

MiniGUI 用 户 手 册

适用于 MiniGUI Ver 1.6.x

北京飞漫软件技术有限公司

2004 年 7 月

简　介

MiniGUI 是由北京飞漫软件技术有限公司拥有版权并主持和维护的自由软件，遵循 GPL 条款发布（1.2.6 及之前的版本以 LGPL 条款发布），其目标是为实时嵌入式 Linux/uClinux、eCos、uC/OS-II、VxWorks 等操作系统建立一个快速、稳定和轻量级的图形用户界面支持系统。

Mde 是 MiniGUI 的综合演示程序包，遵循 GPL 条款发布。

Mg-samples 是 MiniGUI 的示例程序包，遵循 GPL 条款发布。

GNU GPL 条款的原文，可见本手册附录 I。如果您要采用 MiniGUI 开发专有或商用应用软件，则必须首先获得飞漫软件的商业授权。附录 I 给出了飞漫软件针对 MiniGUI 的商业授权策略。

目前，MiniGUI 可免费下载的稳定版本是 1.3.3。您可以从北京飞漫软件技术有限公司网站的“下载”区（http://www.minigui.com/download/cindex.shtml）下载 MiniGUI 源代码、开发文档及示例程序。需要注意的是：免费下载并不意味着免费使用。免费下载的 MiniGUI 只能用于开发 GPL 或其它公开源码的应用软件，如果您要利用 MiniGUI 开发专有或商业软件，则必须从北京飞漫软件技术有限公司购买商业授权。飞漫软件将为购买 MiniGUI 商业授权的用户提供最新版本的 MiniGUI（即 MiniGUI V1.6.x），并提供相关的移植及开发技术支持服务。

本手册详细讲述了 MiniGUI V1.6.x 在 Linux/uClinux 操作系统上的配置、编译、安装及运行，其中还整理罗列了大量常见问题及解决办法，相信读者能够从本书中获得一些有价值的资料，从而帮助用户迅速掌握 MiniGUI 的安装和配置方法，为尽快进入 MiniGUI 应用开发铺平道路。

版权声明

目 录

1 MiniGUI 介绍

1.1 MiniGUI 简介

MiniGUI（http://www.minigui.com）原是由魏永明主持和开发的一个自由软件项目，现由北京飞漫软件技术有限公司维护并开展后续开发。飞漫软件以 GPL 条款的形式发布 MiniGUI，也为专有及商用软件开发者提供商业授权。MiniGUI 项目的最初目标是为基于 Linux 的实时嵌入式系统提供一个轻量级的图形用户界面支持系统。该项目自 1998 年底开始到现在，已历经五年多的开发和应用过程，到目前为止，已经非常成熟和稳定，并且在许多实际产品或项目中得到了广泛应用。另外，现在 MiniGUI 所支持的操作系统不仅仅限于 Linux，它还可以在 uClinux、eCos、uC/OS-II、VxWorks 等操作系统上运行。

MiniGUI 为应用程序定义了一组轻量级的窗口和图形设备接口。利用这些接口，每个应用程序可以建立多个主窗口，然后在这些主窗口中创建按钮、编辑框等控件。MiniGUI 还为用户提供了丰富的图形功能，帮助您显示各种格式的位图并在窗口中绘制复杂图形。

和 Linux 这样的类 UNIX 操作系统相比，一般意义上的嵌入式操作系统具有一些特殊性。举例而言，诸如 uClinux、uC/OS-II、eCos、VxWorks 等操作系统，通常运行在没有 MMU（内存管理单元，用于提供虚拟内存支持）的 CPU 上；这时，往往就没有进程的概念，而只有线程或者任务的概念，这样，GUI 系统的运行环境也就大相径庭。因此，为了适合不同的操作系统环境，我们可将 MiniGUI 配置成三种运行模式：

- MiniGUI-Threads。运行在 MiniGUI-Threads 上的程序可以在不同的线程中建立多个窗口，但所有的窗口在一个进程或者地址空间中运行。这种运行模式非常适合于大多数传统意义上的嵌入式操作系统，比如 uC/OS-II、eCos、VxWorks、pSOS 等等。当然，在 Linux 和 uClinux 上，MiniGUI 也能以 MiniGUI-Threads 的模式运行。

- MiniGUI-Lite。和 MiniGUI-Threads 相反，MiniGUI-Lite 上的每个程序是单独的进程，每个进程也可以建立多个窗口。MiniGUI-Lite 适合于具有完整 UNIX 特性的嵌入式操作系统，比如嵌入式 Linux。

- MiniGUI-Standalone。这种运行模式下，MiniGUI 可以以独立进程的方式运行，既不需要多线程也不需要多进程的支持，这种运行模式适合功能单一的应用场合。比如在一些使用 uClinux 的嵌入式产品中，因为各种原因而缺少线程支持，这时，就可以使用 MiniGUI-Standalone 来开发应用软件。

一般而言，MiniGUI-Standalone 模式的适应面最广，可以支持几乎所有的操作系统，甚至包括类似 DOS 这样的操作系统；MiniGUI-Threads 模式的适用面次之，可运行在支持多任务的实时嵌入式操作系统，或者具备完整 UNIX 特性的普通操作系统；MiniGUI-Lite 模式的适用面较小，它仅适合于具备完整 UNIX 特性的普通操作系统。

但不论采用哪种运行模式，MiniGUI 为上层应用软件提供了最大程度上的一致性；只有少数几个涉及初始化的接口在不同运行模式上有所不同。

附录 A 详细对比了 MiniGUI 三种运行模式的区别及典型应用场合。

目前，MiniGUI 的最新稳定版是版本 1.6.0。您可以从飞漫软件主页的“下载”区（http://www.minigui.com/download/cindex.shtml）下载到 V1.3.3 的源代码。

【注意】在专有及商用产品（即非 GPL 软件）中使用免费下载的 MiniGUI，仍需支付商业授权费用。

MiniGUI 增值版（内含 MiniGUI V1.6.x 或以上版本）是飞漫软件为专有或商用产品开发商提供的 MiniGUI 商业产品。和 MiniGUI 免费下载版本（V1.3.3）相比，除修正了若干缺陷外，MiniGUI 增值版产品含有如下更多功能特性：

- 更多操作系统支持。包括普通嵌入式 Linux、uClinux、eCos、uC/OS-II，以及传统的实时嵌入式操作系统，如 VxWorks。
- 提供了高级二维绘图函数，可以设置线宽、线型、填充模式等。
- 更多硬件开发板支持。包括 Intel DMG 机顶盒、Sigma Designs RealMagic 机顶盒、S3C2410 开发板、华恒 ARM2410 开发板、WinBond SVPxx 可视电话等等。
- 提供了基于皮肤技术的用户界面接口，可帮助您编写华丽的用户界面。
- 更多控件支持。包括透明控件支持，ScrollView 和 ScrollWnd 控件、增强编辑框控件、ListView 控件及网格控件等。
- 我们可以将 MiniGUI 所使用的资源，诸如位图、图标和字体等编译到函数库中，这样 MiniGUI 的运行就不需要在文件系统中单独存放资源文件了，包括 MiniGUI.cfg 配置文件在内，而且将提高 MiniGUI 的初始化速度。这个特性非常适合 uClinux/eCos/uCOS-II 等小型嵌入式操作系统。
- 我们可以将 MiniGUI 配置成以单进程形式运行的版本，也就是说，如果您的系统足够简单，则可以不需要线程支持或者 mginit 程序。
- 我们针对 uClinux 等小型嵌入式操作系统优化了 MiniGUI 的内存使用。
- 我们现在可以使用 make menuconfig 来配置 MiniGUI，从而提供给您非常便利的

配置和定制界面。

本用户手册针对 MiniGUI 1.6.x for Linux/uClinux 增值版编写。

1.2 MiniGUI 的主要功能特征

MiniGUI V1.6.x 的主要特征如下：

1) 提供了完备的多窗口机制和消息传递机制。
2) 提供常用的控件类，包括静态文本框、按钮、单行和多行编辑框、列表框、组合框、进度条、属性页、工具栏、拖动条、树型控件、月历控件等。
3) 对话框和消息框支持。
4) 其它 GUI 元素，包括菜单、加速键、插入符、定时器等。
5) 界面皮肤支持。用户可通过皮肤支持获得外观非常华丽的图形界面。
6) 通过两种不同的内部软件结构支持低端显示设备（比如单色 LCD）和高端显示设备（比如彩色显示器），后者在前者的基础上提供了更加强大的图形功能。
7) Windows 的资源文件支持，如位图、图标、光标等。
8) 各种流行图像文件的支持，包括 JPEG、GIF、PNG、TGA、BMP 等等。
9) 多字符集和多字体支持，目前支持 ISO8859-1～ISO8859-15、GB2312、GBK、GB18030、BIG5、EUC-JP、Shift-JIS、EUC-KR、UNICODE 等字符集，支持等宽点阵字体、变宽点阵字体、Qt/Embedded 使用的嵌入式字体 QPF、TrueType 以及 Adobe Type1 等矢量字体。
10) 多种键盘布局的支持。MiniGUI 除支持常见的美式 PC 键盘布局之外，还支持法语、德语等西欧语种的键盘布局。
11) 汉字（GB2312）输入法支持，包括内码、全拼、智能拼音等。用户还可以从飞漫软件获得五笔、自然码等输入法支持。
12) 针对嵌入式系统的特殊支持，包括一般性的 I/O 流操作，字节序相关函数等。
13) 层的支持。您可以使用 JoinLayer 将一个客户程序加入到某个已由其它客户程序创建好的层中。如果成功，则处于同一个层中的客户能够同时向屏幕上进行图形输出 (该功能增加在 MiniGUI-Lite 版本中)。
14) 借鉴著名的跨平台游戏和多媒体函数库 SDL (Simple DirectMedia Layer) 的新 GAL 接口即 NEWGAL。提供了快速和增强的位块操作，视频加速支持以及 Alpha 混合等功能。
15) 增强的新 GDI 函数，包括光栅操作、复杂区域处理、椭圆、圆弧、多边形以及区域填充等函数。在提供数学库的平台上，还提供有高级二维绘图函数，可设置线宽、线型以及填充模式等。
16) 图形抽象层（GAL）以及输入抽象层（IAL）。利用 GAL 和 IAL，MiniGUI 可以在

许多图形引擎上运行，并且可以非常方便地将 MiniGUI 移植到其他 POSIX 系统上，而这只需要根据我们的抽象层接口实现新的图形引擎即可。目前，我们已经编写了基于 VESA FrameBuffer、QVFB、eCos LCD 的图形引擎，并内建有针对 Xcopilot 仿真器、EP7312 开发板、iPAQ 系列、S3C2410 开发板、Intel DMG 机顶盒等等硬件的输入引擎。利用 QVFB，MiniGUI 应用程序可以运行在 X Window 上，这将大大方便应用程序的调试。

另外，在 MiniGUI 五年的发展过程中，有许多值得一提的技术创新点。正是由于这些技术上的创新，才使得 MiniGUI 更加适合实时嵌入式系统，而且也使得 MiniGUI 具有非常好的灵活性，可以应用在包括手持设备、机顶盒、游戏终端等等在内的各种高端或者低端的嵌入式系统当中。这些技术创新包括：

- 图形和输入抽象层。图形和输入抽象层对顶层 API 没有任何影响，但大大方便了 MiniGUI 自身以及应用程序的移植、调试等工作。MiniGUI 现在已经被证明能够在基于 i386、ARM、MIPS、StrongARM、xScale 以及 PowerPC 等 CPU 的嵌入式系统上流畅运行。
- 多字体和多字符集支持。这部分通过设备上下文（DC）的逻辑字体（LOGFONT）实现，不管是字体类型还是字符集，都可以非常方便地进行扩充。使用 DrawText 等函数时，可通过指定字体而获得对各种字符集支持，比如 GB2312、GBK、GB18030、BIG5、EUCKR、EUCJP、Shift-JIS 等。对于一个窗口来说，同时显示不同语种的文字是可能的。MiniGUI 的这种字符集支持不同于通过 UNICODE 实现的传统多字符集支持，这种实现占用资源少，更加适合于嵌入式系统。
- 多种运行模式。MiniGUI-Threads、MiniGUI-Lite 及 MiniGUI-Standalone 等三种运行模式大大扩展了 MiniGUI 适应各种嵌入式操作系统的能力，也为移植 MiniGUI 应用程序到不同的操作系统上提供了便利。
- 利用 MiniGUI 的 NEWGAL 接口，应用程序不仅能够获得高级及扩展的图形绘制功能，而且大大增强了 MiniGUI 适应各种不同显示设备的能力。利用 MiniGUI 的 NEWGAL 接口，MiniGUI 不仅可以支持提供有直接帧缓冲区的设备，也可以支持通过 OSD 以及其它特殊硬件接口显示图形的设备。

MiniGUI 新版本在图形方面的增强和提高，将大大扩展它的应用领域，从而能够为嵌入式系统上的多媒体应用、游戏开发等提供支持。

1.3 MiniGUI 的优势

在实时嵌入式操作系统中，其硬件环境比较苛刻，因此要求运行其中的图形界面尽可能的精简，而传统的窗口系统尚不能满足实时嵌入式系统的需求。所以，在实时嵌入式系统上，

开发一个能够充分满足嵌入式系统需求的图形用户界面支持系统就成了当务之急。实际上，国内外已有许多专门针对 Linux 或其它嵌入式操作系统的嵌入式 GUI 系统，MiniGUI 只是其中之一。然而，由于开发人员对实时嵌入式系统在理解上的不同，使得这些 GUI 系统在接口定义、体系结构、功能特性等方面存在着很大的差别。另外，这些 GUI 系统所使用的授权条款也各有不同。我们的开发目标是为所有的中端和高端的智能信息设备提供稳定、高性能的用户图形系统。

比较目前几个面向实时嵌入式 Linux 系统的 GUI，我们认为目前比较成熟，同时得到最多开发人员认可的有 MicroWindows、Qt/Embedded 以及 MiniGUI 等三个系统。相比较而言，MiniGUI 在如下几个方面有其独特的优势。

1.3.1 轻型、占用资源少

MiniGUI 本身的占用空间非常小。以嵌入式 Linux 操作系统为例，MiniGUI 的典型存储空间占用情况如下：

- Linux 内核： 300K ～ 500K（由系统决定）
- MiniGUI 支持库： 500K ～ 700K（由编译选项确定）
- MiniGUI 字体、位图等资源： 400K （由应用程序确定，可缩小到 200K 以内）
- GB2312 输入法码表：200K（不是必需的，由应用程序确定）
- 应用程序： 1M ～ 2M（由系统决定）

总体的系统占有空间应该在 2MB 到 4MB 左右。在某些系统上，功能完备的 MiniGUI 系统本身所占用的空间可进一步缩小到 1MB 以内。

MiniGUI 最初是为了满足一个工业控制系统（计算机数控系统）的需求而设计和开发的。这个工业控制系统是清华大学为一台数控机床设计的计算机数控系统（CNC）。在比较 DOS、Windows 98、Windows NT、Linux 等系统之后，该项目组决定选择 RT-Linux 作为实时操作系统，以便满足 2ms 甚至更高的实时性。但是图形用户界面是一个问题，因为 X Window 不适合于实时控制系统，并且当时 X Window 系统的本地化也不尽人意。因此，决定自己开发一套图形用户界面支持系统。这就是 MiniGUI 产生的背景。显然，MiniGUI 一开始就针对实时系统而设计，因此，在设计之初就考虑到了小巧、高性能和高效率。目前，这个数控系统的开发已经完成，MiniGUI 在其中担当了非常重要的角色。

最新的研发成果表明，MiniGUI 能够在 CPU 主频为 30 MHz，仅有 4M RAM 的系统上正常运行（使用 uClinux 操作系统），这是其它针对嵌入式产品的图形系统，比如 MicroWindows 或者 Qt/Embedded 所无法达到的。

1.3.2 高性能

在比较上述三种 GUI 系统的性能时，任何一个人单从肉眼就能够看出在图形方面的效率高低。MicroWindows 追求和 X 的兼容，所以，采用的传统的基于 UNIX 套接字的客户/服务器系统结构。在这种体系结构下，客户建立窗口、绘制等等都要通过套接字传递到服务器，由服务器完成实质工作。这样，系统非常依赖于 UNIX 套接字通讯；而大家都知道，UNIX 套接字的数据传递，要经过内核，然后再传递到另外一个程序。这样，大量的数据在客户/内核/服务器之间传递，从而增加了系统负荷，也占用了许多系统资源。加上 MicroWindows 的图形引擎代码未经任何优化，因此，MicroWindows 的图形效率很低。

Qt/Embedded 是 C++ 的函数库，影响其图形效率的原因，主要是 C++ 的臃肿和由此造成的系统资源的极度浪费。从对用户操作的响应能力、应用程序的启动速度等方面看，Qt/Embedded 的速度是最慢的。MiniGUI 为提高整体性能，首先采用了独特的体系结构，其次对图形系统进行了大规模的优化。

MiniGUI 良好的体系结构及优化的图形接口，可确保最快的图形绘制速度。在设计之初，我们就充分考虑到了实时嵌入式系统的特点，针对多窗口环境下的图形绘制开展了大量的研究及开发，优化了 MiniGUI 的图形绘制性能及资源占有。MiniGUI 在大量实际系统中的应用，尤其在工业控制系统的应用，证明 MiniGUI 具有非常好的性能。

1.3.3 高可靠性

从 1999 年 MiniGUI 的第一个版本发布以来，就有许多产品和项目使用 MiniGUI，MiniGUI 本身也不断从这些产品或者项目当中获得发展动力和新的技术需求，不断提高了自身的可靠性和健壮性。

有关 MiniGUI 的最新成功案例，请访问：

```
http://www.minigui.com/project/cindex.shtml
```

1.3.4 可伸缩性强

MiniGUI 丰富的功能和可配置特性，使得它既可运行于基于龙珠的低端产品中，亦可运行于基于 ARM9 的高端产品中，并使用 MiniGUI 的高级控件风格及皮肤界面等技术，创建华丽的用户界面。

1.3.5 跨操作系统支持

理论上，MiniGUI 可支持任意一个多任务嵌入式操作系统；实际已支持 Linux/uClinux、eCos、uC/OS-II、VxWorks 等嵌入式操作系统。同时，在不同操作系统上的 MiniGUI，

提供完全兼容的 API 接口。有关 MiniGUI 和嵌入式操作系统的关系，可参阅本手册附录 E。

1.3.6 可配置

为满足嵌入式系统千变万化的需求，必须要求 GUI 系统是可配置的。在 CNC 系统中得到成功应用之后，我们立即着手于 MiniGUI 可配置的设计。我们通过 Linux 下的 Automake 和 Autoconf 接口，实现了大量的编译配置选项，通过这些选项可指定 MiniGUI 库中包括哪些功能而同时不包括哪些功能。大体说来，您可以在如下几个方面对 MiniGUI 进行定制配置：

- 指定 MiniGUI 要运行的操作系统，是普通嵌入式 Linux、uClinux、eCos 还是 uC/OS-II 或者 VxWorks。
- 指定生成基于线程的 MiniGUI-Threads 运行模式还是基于进程的 MiniGUI-Lite 运行模式，或者只是最简单的 MiniGUI-Standalone 运行模式。
- 指定要采用老的 GAL/GDI 接口还是新的 GAL/GDI 接口。
- 指定需要支持的 GAL 引擎和 IAL 引擎，以及引擎相关选项。
- 指定需要支持的字体类型。
- 指定需要支持的字符集。
- 指定需要支持的图像文件格式。
- 指定需要支持的控件类。
- 指定控件的整体风格，是三维风格还是平面风格等。

总之，MiniGUI 是一个非常适合于工业控制实时系统以及嵌入式系统的高效、可靠、可定制、小巧灵活的图形用户界面支持系统。

1.4 其它信息

MiniGUI 增值版产品中还包含有如下相关手册：

- 《MiniGUI 编程指南》 V1.6 For MiniGUI 1.6.x。描述利用 MiniGUI 开发应用程序时的相关概念、主要的 API 接口及示例程序等。
- 《MiniGUI API Reference Manual》 V1.6 For MiniGUI 1.6.x。对 MiniGUI API 接口的详细描述。

在 Linux 上运行 MiniGUI 及其应用程序，可帮助您尽快掌握 MiniGUI 配置、编程及相关的知识。同时，您可以在 Linux 操作系统上将 MiniGUI 配置成上面提到的三种运行模式中的任何一种。因为除了和具体设备相关的程序之外，大部分基于 MiniGUI 的应用程序

均可以在 Linux 上编写并调试，然后通过交叉编译下载到目标板上运行，因此，我们建议您在使用 MiniGUI 开发嵌入式产品之前，准备好一台安装有 Linux 操作系统的 PC 机，并在 Linux 环境下完成图形相关的开发及测试，最后在目标板上测试并运行。我们推荐运行 Linux 的 PC 机配置如下：

- 奔腾以上 CPU；
- 64MB 以上内存；
- 至少 4GB 空闲硬盘空间；
- 使用 USB/PS2 接口的鼠标（PS2 鼠标协议）；
- VESA2 兼容的显示卡，确保能达到 1024x768 分辨率，16 位色。
- 安装 Red Hat Linux 9，并选择 GNU 开发工具链、X Window、GNOME 等常用软件。

本手册重点讲述 MiniGUI 在 Linux/uClinux 操作系统上的运行，同时以 uClinux 及 Xcopilot 模拟器为例，讲述了将 MiniGUI 及应用程序移植到该模拟器上的一般过程。将 MiniGUI 及基于 MiniGUI 的应用程序交叉编译并在目标板上运行的方法，随着 MiniGUI 所运行的操作系统及目标板的不同而不同。MiniGUI 在其它操作系统及目标板的编译、安装及运行由相应文档描述。如果您购买的 MiniGUI 增值版来自飞漫软件的合作伙伴，则增值版产品中已包含有针对特定目标板及操作系统的说明文档（如《MiniGUI 在 UP-NETARM3000-uClinux 开发板上的使用指南》）；关于特定目标板及操作系统上的 MiniGUI 使用，请参阅对应的文档。

请访问 http://www.minigui.com/product/cindex.shtml 获得 MiniGUI 增值版及其它产品的购买信息。

2 快速开始

假定您第一次安装 MiniGUI，本章将引导您在自己运行有 Linux 的 PC 机上以默认方式快速安装并运行 MiniGUI。

2.1 设置 MiniGUI 的 PC 运行环境

我们假定用户使用的 Linux 系统是 RedHat Linux 6.x 及以上发行版，使用 Linux 内核 2.2.x、2.4.x 或者 2.6.x。为了运行 MiniGUI，我们需要激活系统中的 FrameBuffer 设备驱动程序。对大部分使用兼容 VESA 标准的显示卡的 PC 机来讲，可以遵循下面的步骤：

1) 确保您的 PC 机显示卡是 VESA 兼容的。大多数显示卡是 VESA 兼容的，然而某些内嵌在主板上的显示卡可能不是 VESA 兼容的，比如 Intel i810 系列。如果显示卡是 VESA 兼容的，就可以使用 Linux 内核中的 VESA FrameBuffer 驱动程序了。

2) 确保您的 Linux 内核包含了 FrameBuffer 支持，并包含了 VESA FrameBuffer 驱动程序。RedHat Linux 6.x 及以上的发行版自带的内核中已经包含了该驱动程序。如果使用自己编译的内核，请检查您的内核配置。

3) 如果使用 LILO 引导装载器，则需要修改 /etc/lilo.conf 文件，在您所使用的内核选项段中，添加如下一行（使用 GRUB 的用户请转到第 6 步）：

```
    vga=0x0317
```

这样，Linux 内核在启动时将把显示模式设置为 1024x768x16bpp 模式。如果您的显示器无法达到这种显示分辨率，可考虑设置 vga=0x0314，它对应 800x600x16bpp 显示模式。修改后的 /etc/lilo.conf 文件类似：

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
linear
default=linux

image=/boot/vmlinuz-2.4.2
    vga=0x0317          ; 这一行设置显示模式.
    label=linux
    read-only
    root=/dev/hda6

other=/dev/hda1
    label=dos
```

4) 运行 lilo 命令，使所作的修改生效，并重新启动系统：

```
# lilo
# reboot
```

5) 如果一切正常，将在 Linux 内核的引导过程中看到屏幕左上角出现可爱的 Linux 吉祥物——企鹅，或者 RedHat Linux 的蓝天白云产品徽标，并发现系统的显示模式发生了变化。

6) 如果读者使用的是 Red Hat 7.x 或者更高版本，并且在安装 Red Hat 时使用了 GRUB 而不是 LILO 作为引导装载器，则设置 FrameBuffer 的方法会有一些不同：

第一，要激活 VESA FrameBuffer 驱动程序，需要修改 /boot/grub/menu.lst 文件，并在 kernel 打头的一行添加 vga=0x0317。您也可以复制已有的引导选项并修改复制之后的选项，例如：

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,0)
#          kernel /boot/vmlinuz-version ro root=/dev/hda1
#          initrd /boot/initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.4.18-3, FrameBuffer)
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hda1 vga=0x0317
        initrd /boot/initrd-2.4.18-3.img

title Red Hat Linux (2.4.18-3)
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hda1
        initrd /boot/initrd-2.4.18-3.img
```

其中 Red Hat Linux (2.4.18-3, FrameBuffer) 就是设置了 VESA FrameBuffer 的引导选项。

第二，修改了 /boot/grub/menu.lst 文件之后，重新启动系统即可，而无需执行类似 lilo 那样的命令。

在正确激活 VESA FrameBuffer 设备驱动程序之后，我们就可以开始安装并运行 MiniGUI 了。如果因为某种原因，您的系统无法激活 VESA FrameBuffer，则可以参考本手册附录 C“建立 MiniGUI 运行环境”使用其他的图形引擎来运行 MiniGUI。

2.2 解开 MiniGUI 源代码及资源包

MiniGUI 增值版用户可直接从产品光盘上获得最新的 1.6.x 版本。MiniGUI 相关的文件如下所列：

- libminigui-1.6.x.tar.gz：MiniGUI 函数库源代码，其中包括 libminigui、libmgext 和 libvcongui。
- minigui-res-1.6.x.tar.gz：MiniGUI 所使用的资源， 包括基本字体、图标、位图和鼠标光标。
- mde-1.6.x.tar.gz：MiniGUI 的综合演示程序。
- mg-smaples-1.6.x.tar.gz：《MiniGUI 编程指南》的配套示例程序。

上述文件保存在产品 CD-ROM 的 /minigui/1.6.x/ 目录下。

2.3 安装资源文件

我们首先要安装 MiniGUI 的资源文件。请按照如下步骤：

1) 使用 tar 命令解开 minigui-res-1.6.x.tar.gz，可使用如下命令：

```
$ tar zxf minigui-res-1.6.x.tar.gz
```

2) 该命令将建立 minigui-res-1.6.x/ 目录。用 cd 命令改变到新建目录中，然后以超级用户身份运行 make install 命令：

```
$ su -c 'make install'
```

或者

```
$ su
# make install
```

2.4 配置和编译 MiniGUI

MiniGUI 使用了自由软件常用的“automake”和“autoconf”接口，因而，MiniGUI 在 GNU 开发环境下的配置和编译非常容易：

1) 使用 tar 解开 libminigui-1.6.x.tar.gz 到新的目录：

```
$ tar zxf libminigui-1.6.x.tar.gz
```

2) 该命令将建立 libminigui-1.6.x/ 目录。改变到这一新目录，然后运行 ./configure：

```
$ ./configure
```

3) 运行下面的命令编译并安装 MiniGUI：

```
$ make; su -c 'make install'
```

4) 默认情况下，MiniGUI 的函数库将安装在 /usr/local/lib 目录中。您应该确保该目录已经列在 /etc/ld.so.conf 文件中。修改 /etc/ld.so.conf 文件，将 /usr/local/lib 目录添加到该文件最后一行。修改后类似：

```
/usr/lib
/usr/X11R6/lib
/usr/i486-linux-libc5/lib
/usr/local/lib
```

5) 安装 MiniGUI 之后，运行下面的命令更新共享函数库系统的缓存：

```
$ su -c /sbin/ldconfig
```

2.5 编译并运行 MiniGUI 的演示程序

在安装好 MiniGUI 函数库之后，应该解开并编译 MDE 演示程序包：

1) 使用 tar 命令将 mde-1.6.x.tar.gz 软件包解开到新的目录：

```
$ tar zxf mde-1.6.x.tar.gz
```

2) 进入 mde-1.6.x/ 目录，依次运行 ./configure 和 make 编译演示程序：

```
$ cd mde-1.6.x
$ ./configure
$ make
```

3) 如果编译过程中出现错误，则请检查 2.4 中的步骤。若没有出现错误，则可以进入 mginit/ 目录启动 MiniGUI-Threads 版本的程序：

```
$ cd gdidemo
$ ./gdidemo
```

该程序将启动字体演示程序。

图 2-1 是 gdidemo 命令运行起来的屏幕图。

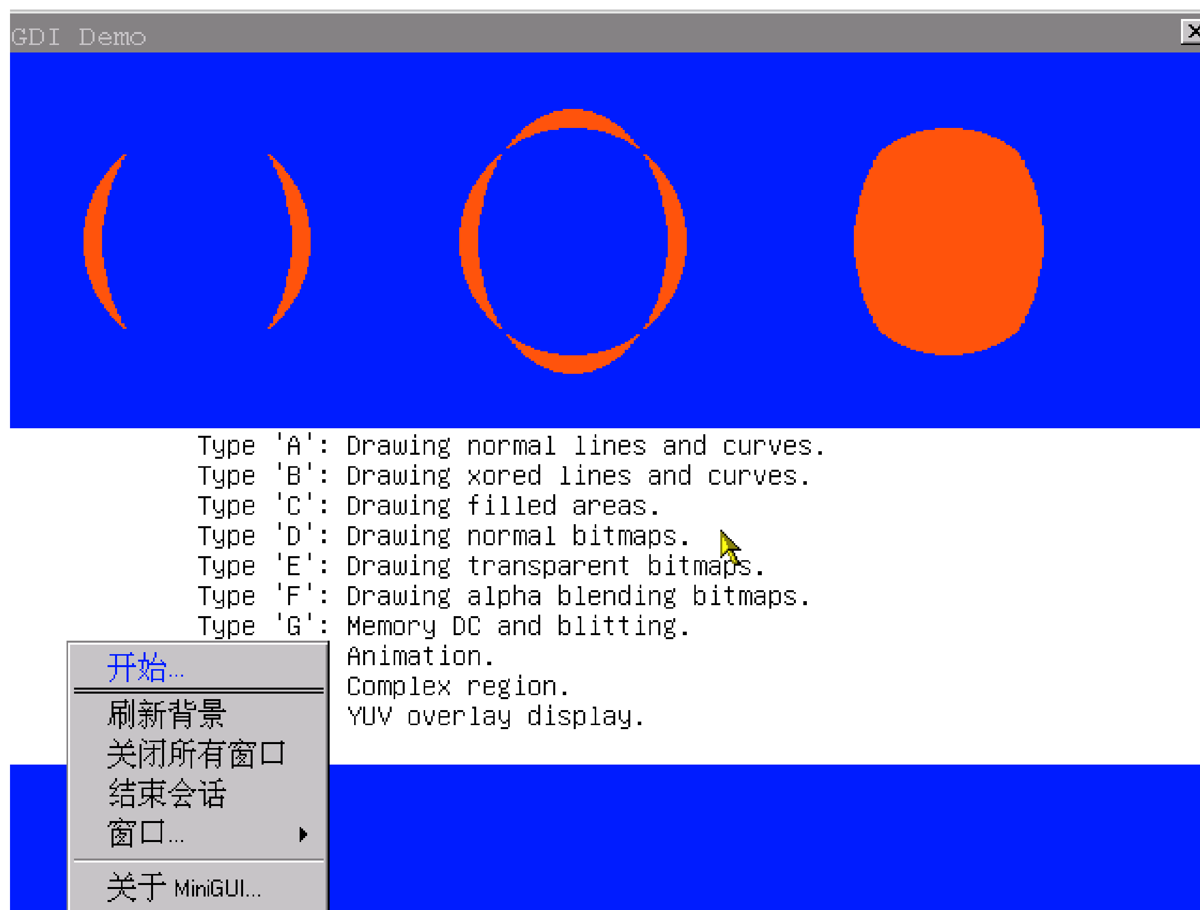


图 2-1 运行 gdidemo 程序

【注意】该手册给出的屏幕截图是 **MiniGUI 1.6.x** 中的 **MDE** 程序运行效果。使用免费下载的版本，因为所使用字体不同，显示效果会有较大差别。

4) 在程序桌面上点击鼠标右键，然后从菜单中选择“关闭所有窗口”，之后再次点击鼠标邮件并选择“结束回话”，即可退出 MiniGUI。您也可以在任何时候按下 <Ctrl+Alt+BackSpace> 键强制退出 MiniGUI。

【提示】**MiniGUI** 演示程序默认在已激活 **FrameBuffer Linux** 控制台上运行。如果您的 **Linux** 控制台未激活 **FrameBuffer** 驱动程序，则可参阅附录 **C.2** 在 **X Window** 上利用 **QVFB** 程序运行。

2.6 MiniGUI 和嵌入式系统

为了更好地帮助用户在自己的嵌入式系统上运行 MiniGUI，我们在 MiniGUI 增值版产品中提供了对基于 Linux/uClinux 操作系统的嵌入式系统支持：

- 通常，我们在 PC 机上开发 MiniGUI 应用程序，然后将 MiniGUI 及自己的应用程

序移植到目标系统。这其中涉及到一个重要的步骤——交叉编译。MiniGUI 增值版产品中包含了常见嵌入式 CPU 的交叉编译工具，其中包括 ARM、PowrPC、MIPS、m68k 等等。本手册第 5 章和第 6 章讲述这些交叉编译工具的安装以及 MiniGUI 和应用程序的移植。

- 增值版产品中还包含有 MiniGUI 所依赖的若干函数库的源代码。我们可以利用该增值版产品中带有交叉编译工具对这些函数库进行交叉编译，以便用于自己的特定硬件，其交叉编译过程可参照本手册第 5 章“MiniGUI 的交叉编译”。

3 在运行 Linux 的 PC 机上安装并运行 MiniGUI

本章将详细讲述 MiniGUI 的在运行 Linux 的 PC 机上的配置、安装及设置。MiniGUI 针对特定操作系统及目标板的配置、安装和设置过程和该过程类似。

3.1 MiniGUI 的组成

MiniGUI 图形系统由函数库、资源、演示程序三部分组成：

1) MiniGUI 的函数库部分由三个函数库组成。它们分别是 libminigui、libmgext 以及 libvcongui。libminigui 是提供窗口管理和图形接口的核心函数库，也提供了大量的标准控件；libmgext 是 libminigui 的一个扩展库，提供了一些高级控件以及“文件打开”对话框等；libvcongui 则提供了一个应用程序可用的虚拟控制台窗口，从而可以方便地在 MiniGUI 环境中运行字符界面的应用程序，libmgext 和 libvcongui 库已经包含在 libminigui-1.6.x.tar.gz 包中。
2) 运行 MiniGUI 所需的资源。这些资源包括运行 MiniGUI 应用程序需要的基本字体、图标、位图以及鼠标光标等。这些资源包含在 minigui-res-1.6.x.tar.gz 软件包中发布。
3) MiniGUI 的演示程序包是 mde-1.6.x.tar.gz。这个包中包含了 MiniGUI 的控件演示程序、字体演示程序、对话框演示程序、记事本等应用程序，还有推箱子、扫雷、俄罗斯方块等游戏。
4) MiniGUI 的示例程序包是 mg-samples-1.6.x.tar.gz。这个包中包含了《MiniGUI 编程指南》所使用示例程序。

3.2 如何获得 MiniGUI

您可以从北京飞漫软件技术有限公司网站的“下载”区（http://www.minigui.com/download/cindex.shtml）下载得到 MiniGUI V1.3.3 版本以及对应的资源和演示程序包。免费下载获得的版本，只能用于开发 GPL 应用软件，如果您要开发专有或商用产品，则必须获得飞漫软件的商业授权。

北京飞漫软件技术有限公司购买 MiniGUI 商业授权的用户提供增值版产品，具体可参阅飞漫软件公司网站的“产品”区（http://www.minigui.com/product/cindex.shtml）。

【注意】北京飞漫软件技术有限公司遵循 **GPL** 条款发布 **MiniGUI1.6.x** 版本的三个函数库；遵循 **GPL** 条款发布 **MDE** 演示程序及 **MG-SAMPLES** 示例程序；**minigui-res** 等资源包中的文件，仅限于配合 **MiniGUI** 使用，不得用

于其它场合。

3.3 建立 MiniGUI 运行环境的前提

运行 MiniGUI 的系统需要满足一些前提条件：

- 支持 POSIX1.X 的 UNIX 系统。这包括 Linux 2.0、2.2 和 2.4 等，也包括 uClinux 等非标准 Linux 系统。MiniGUI 1.6.x 亦可在 eCos、uC/OS-II 及 VxWorks 操作系统上运行。
- 如果在 Linux/uClinux 上运行，要求 Linux FrameBuffer 驱动程序的功能正常。在 PC 上，如果显示芯片是 VESA 兼容的，则可以通过 Linux 的 VESA FrameBuffer 驱动程序获得较好的支持。对没有 FrameBuffer 支持的 Linux/uClinux 系统，需要编写特定的图形引擎才能运行 MiniGUI。MiniGUI 1.6.x 中已包含了针对非 FrameBuffer 驱动的图形引擎实现，比如在 EM85xx 板上的 OSD 图形接口。
- MiniGUI-Lite 的运行需要系统提供 System V 的如下进程间通讯机制：共享内存和信号量。
- 运行 MiniGUI-Lite 需要 UNIX Domain 套接字机制的支持。
- 运行 MiniGUI-Threads 需要 POSIX 兼容线程库的支持。

以上条件可用来指导嵌入式 Linux/uClinux 或者其它操作系统的选择和定制。

3.4 MiniGUI 的编译和安装

3.4.1 Linux 下的软件维护和建立工具

对开放源码的自由软件来说，程序员得到的通常是源代码，在编译源代码并正确安装和配置的过程中，往往会涉及到许多工具和函数库，因此其过程经常显得有些繁复。MiniGUI 也不例外。为了说明 MiniGUI 的正确编译和安装过程，我们有必要首先了解 Linux 系统下用于软件维护和建立的工具。

1) make 和 makefile

make 是 Linux 下最常用的二进制程序、函数库的建立生成工具。make 运行时要根据当前目录下的 makefile 文件（一般是 Makefile），确定要生成什么样的二进制文件，以及对应的命令。我们还可以在 makefile 文件中建立要生成的目标与源代码之间的依赖关系，从而可以让 make 工具根据时间自动判断是否需要通过中间过程而生成最终目标。尽管通过 makefile 文件可以组织一个大的项目，但往往手工编写一个 makefile 文件并不是一件轻松的事情，并且在需要维护一个源代码的目录树时，makefile 文件的维护工作就会大大

增加。为此，GNU 又开发了 Autoconfi/Automake 工具，可以用来自动生成 makefile 文件，并且能够检查系统的配置信息，从而帮助提供源代码的可移植性。

2) Autoconf/Automake

GNU 的 Autoconf 及 Automake 这两个软件实际是由若干 Shell 脚本组成的，它可以帮助程序员轻松产生 makefile 文件。现在的各种自由软件，如 Apache、MySQL 等都是利用 Autoconf、Automake 实现自动配置和编译的。MiniGUI 也采用了 Autoconf/Automake 接口。用户只要使用“./configure”、“make”、“make install” 三条命令就可以把程序或函数库编译并安装到系统中。

利用 configure 所产生的 Makefile 文件有几个预先设定的目标可供使用，这里只对其中几个简述如下：

- make all 产生设定的目标。只敲入 make 也可以，此时会开始编译源代码，然后连接并产生执行文件。
- make clean 清除之前所编译的可执行文件及目标文件（*.o)。
- make distclean 除了清除可执行文件和目标文件以外，也把 configure 所产生的 Makefile 清除掉。通常在发布软件前执行该命令。
- make install 将程序安装到系统中，若源码编译成功，且执行结果正确，便可以把程序安装到系统预先设定的执行文件存放路径中，若用 bin_PROGRAMS 宏的话，程序会被安装到 /usr/local/bin 下。
- make dist 将程序和相关的文档包装为一个压缩包以供发布。执行该命令后，目录下会产生一个以 PACKAGE-VERSION.tar.gz 为名称的文件。PACKAGE 和 VERSION 这两个参数是根据 configure.in 文件中 AM_INIT_AUTOMAKE (PACKAGE, VERSION) 宏定义的。
- make distcheck 和 make dist 类似，但是加入检查打包以后的压缩文件是否正常，这个目标除了把程序和相关文档打包成 tar.gz 文件外，还会自动把这个压缩文件解开，执行 configure，并执行 make all，确认编译无错误以后，方显示这个 tar.gz 文件已经准备好并可以发布了。

要注意的是，利用 Autoconf 及 Automake 所产生出来的软件包可以在没有安装 Autoconf 及 Automake 环境中使用，这是因为 configure 是一个 shell 脚本，它已被设计为可以在一般的 Unix Shell 下执行。

3) ldd 和 ldconfig

ldd 是用来检查可执行文件所需要的共享库。例如：

```
$ ldd /bin/ls
```

```
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x4001c000)
        libc.so.6 => /lib/libc.so.6 (0x40020000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

我们在 /bin/ls 程序上运行 ldd 命令，就可以检查该程序所使用的共享库。注意在 ldd 命令打印的结果中，“=>”左边的表示该程序需要连接的共享库之 so 名称，右边表示由 Linux 的共享库系统找到的对应的共享库在文件系统中的具体位置。默认情况下，/etc/ld.so.conf 文件中包含有默认的共享库搜索路径，例如：

```
/usr/X11R6/lib
/usr/lib
/usr/i486-linux-libc5/lib
/usr/lib/qt-2.0.1/lib
/usr/lib/qt-1.44/lib
/usr/lib/qt-2.1.0/lib
/usr/kerberos/lib
/usr/lib/qt-1.45/lib
```

通常情况下，许多开放源代码的程序或函数库都会默认将自己安装到 /usr/local 目录下的相应位置（/usr/local/bin 或 /usr/local/lib），以便与系统自身的程序或函数库相区别。而许多 Linux 系统的 /etc/ld.so.conf 文件中默认又不包含 /usr/local/lib。因此，往往会出现已经安装了共享库，但是却无法找到共享库的情况。这时，就应该检查 /etc/ld.so.conf 文件，如果其中缺少 /usr/local/lib 目录，就应该添加进去。

在修改了 /etc/ld.so.conf 文件或者在系统中安装了新的函数库之后，还要运行一个命令，即 ldconfig。该命令用来刷新系统的共享库缓存，即 /etc/ld.so.cache 文件。为了减少共享库系统的库搜索时间，共享库系统维护了一个共享库 so 名称的缓存文件。因此，在安装新的共享库之后，一定要运行 ldconfig 刷新该缓存。

3.4.2 MiniGUI 的图形引擎

在 MiniGUI 目前的版本中，主要有五个图形引擎：基于 VESA FrameBuffer 的图形引擎（我们称之为 NATIVE 引擎或者 FBCON 引擎）、SVGALib 引擎、LibGGI 引擎，还有基于 QVFB 的图形引擎以及“哑”图形引擎。其中 NATIVE/FBCON 引擎已经包含在 MiniGUI 本身的代码中，不用额外安装依赖库，这也是我们推荐用户使用的图形引擎。但如果在 Linux 内核 2.0 版本上使用 MiniGUI 时，就需要使用 SVGALib 或者 LibGGI 这两种图形引擎，这时，需要安装必要的依赖函数库，具体的下载和安装指导请参阅附录 B。有关 FBCON 引擎和 QVFB 引擎的配置信息，请参阅附录 C“建立 MiniGUI 的 PC 运行环境”。

3.4.3 MiniGUI 的依赖库

为了正确安装 MiniGUI，还需要了解 MiniGUI 需要哪些函数库，也即 MiniGUI 的依赖库。在编译 MiniGUI 之前，首先要确保正确安装了所需的依赖库。除了在使用 SVGALib 和

LibGGI 图形引擎时需要第三方函数库的支持外，MiniGUI 还使用了其它一些第三方的依赖库。

1) LibTTF 和 LibT1

这两个函数库分别提供对 TrueType 字体和 Adobe Type1 字体的支持。对 MiniGUI 来说，这两个函数库是可选的。如果需要 TrueType 和 Adobe Type1 字体的支持，则需要首先安装这两个函数库，否则就不必安装。MiniGUI 的 configure 脚本可以自动检查系统中是否安装有这两个函数库，如果没有安装，则会取消对 TrueType 和 Adobe Type1 字体的支持。

现在的 Linux 发行版默认会安装 LibTTF（即 FreeType 库）。但是，需要注意的是，MiniGUI 所使用的 LibTTF 是 1.3.1 版本的。如果系统中已安装的 LibTTF 版本不同，则可能因为兼容性问题而无法正确编译 MiniGUI。因此，需要事先确认 LibTTF 的版本号，可通过：

```
$ ls -l /usr/lib/libttf*
```

命令来检查已安装的 LibTTF 库的版本号，其中的共享库文件名中包含了该函数库的版本号。

用户也可以从北京飞漫软件技术有限公司网站的“免费下载”区（http://www.minigui.com/download/c3rdparty.shtml）下载 MiniGUI 所使用的 libttf 和 libt1 库。

2) LibJPEG、LibPNG 等函数库

其他 MiniGUI 所依赖的函数库包括用来支持 JPEG 图片的 libjpeg、用来支持 PNG 图片的 libpng、提供 POSIX 兼容线程支持的 libpthread 等等。一般的 Linux 发行版中均已包含有这些函数库。

3.4.4 编译并安装 MiniGUI

在安装好 MiniGUI 的上述依赖库之后，就可以编译并安装 MiniGUI 了。首先从增值版光盘内复制 MiniGUI 的函数库包、演示程序包和资源包：

- libminigui-1.6.x.tar.gz
- minigui-res-1.6.x.tar.gz
- mde-1.6.x.tar.gz
- mg-samples-1.6.x.tar.gz

【注意】上述文件保存在增值版光盘的 /minigui/1.6.x/ 目录下。

1) 编译并安装 libminigui

用如下命令解开 libminigui-1.6.x.tar.gz 软件包：

```
$ tar zxf libminigui-1.6.x.tar.gz
```

该命令将在当前目录建立 libminigui-1.6.x 目录。进入该目录，并运行 ./configure 命令：

```
$ cd libminigui-1.6.x
$ ./configure
```

不带任何参数执行 ./configure 脚本将按照默认选项生成 Makefile。运行 ./configure --help 可以看到各种可配置的选项，关于可配置选项的具体含义，将在下一节详细介绍。简单说来，运行 ./configure --enable-xxx 可以配置 MiniGUI 函数库打开某项功能，而运行 ./configure --disable-xxx 则可以禁止某项功能。

根据自己的需求确定好 MiniGUI 库中要包含的功能特色之后，就可以运行类似上面的命令生成定制的 Makefile 文件。如果运行 ./configure 脚本的时候没有出现问题，就可以继续运行 make 和 make install 命令编译并安装 libminigui，注意要有 root 权限才能向系统中安装函数库：

```
$ make
$ su -c make install
```

如果在运行 ./configure 命令时出现函数库检查错误，通常是 MiniGUI 的依赖库没有正确安装，而用户却配置 MiniGUI 使用此依赖库。这时，请检查相应依赖库的安装情况。

在一切正常之后，确保已经将 /usr/local/lib 目录添加到 /etc/ld.so.conf 文件中，运行 ldconfig 命令刷新系统的共享库搜索缓存：

```
$ su -c /sbin/ldconfig
```

2) 安装 MiniGUI 的资源

MiniGUI 资源的安装比较简单，只需解开软件包并以 root 身份运行 make install 命令，如下所示：

```
$ tar zxf minigui-res-1.6.x.tar.gz
$ cd minigui-res-1.6.x
$ su -c make install
```

默认的安装脚本会把 MiniGUI 资源文件安装到 /usr/local/lib/minigui/res/ 目录下。

3) 编译并安装 MiniGUI 的演示程序 mde

编译和安装 mde-1.6.x.tar.gz 的过程与 libminigui-1.6.x.tar.gz 类似，所需命令如下：

```
$ tar zxf mde-1.6.x.tar.gz
$ cd mde-1.6.x
$ ./configure
$ make
```

在运行 MDE 的 configure 脚本时，它会根据当前系统中的 MiniGUI 配置作出不同的编译选择；如果您将 MiniGUI 配置成 MiniGUI-Lite，则 MDE 就会编译出 mginit 程序，如果配置成 MiniGUI-Threads 或者 MiniGUI-StandAlone，则 MDE 就不会编译 mginit 程序。

3.4.5 MiniGUI-Threads 的运行

如果您将 MiniGUI 配置成 MiniGUI-Threads 版本，则可以直接运行 MDE 中的应用程序，而无须首先执行 mginit 程序。实际上，在 MiniGUI 配置成 MiniGUI-Threads 时，MDE 根本不会去编译 mginit 程序。

图 3-1 是在 MiniGUI-Threads 上运行 fontdemo 程序的屏幕截图。

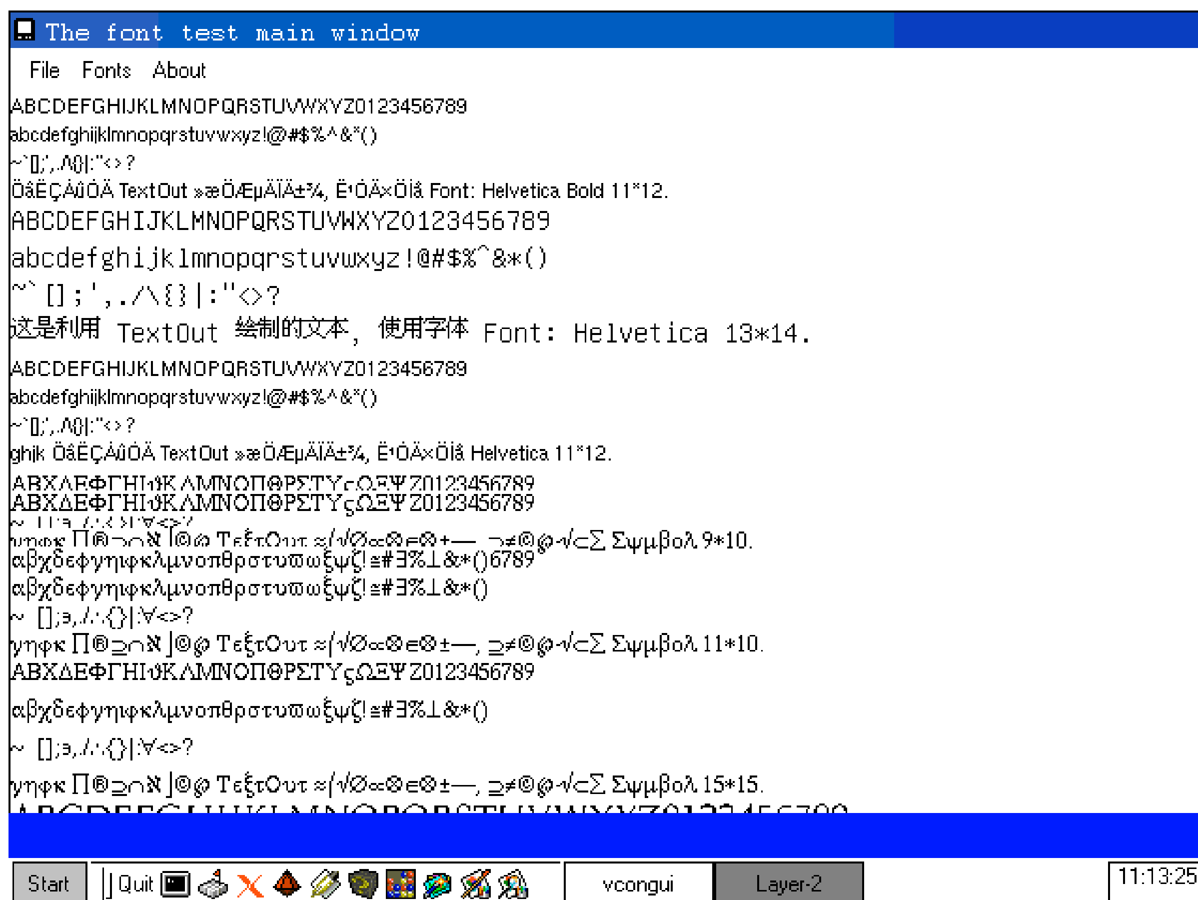


图 3-1 fontdemo 程序

3.4.6 MiniGUI-Lite 的运行

如果您在配置 MiniGUI 时使用 –enable-lite 选项，则 MiniGUI 将以 MiniGUI-Lite 模式运行。编译 MDE 之后，需进入 mde-1.6.x/mginit 目录，然后运行 mginit 程序。

```
$ cd mde-1.6.x/mginit
$ ./mginit
```

如果在运行过程中出现错误，首先可运行 ldd mginit 命令检查 Linux 是否正确找到了 mginit 程序所需要的共享库，若没有找到，则需要检查是否设置了正确的共享库搜索路径并运行了 ldconfig 命令。若出现其他问题，请参阅第 4 章“MiniGUI 的配置”和附录 H“常见错误及解答”。

如果正确安装了 MiniGUI 函数库和资源文件，这时就可以看到 MiniGUI 演示程序的画面了。还可以在 MiniGUI 的虚拟控制台程序中运行 mde 目录下的各个子目录下的演示程序。比如 dlgdemo、ctrls、fontdemo 等。

图 3-2 是在 MiniGUI-Lite 上运行 dlgdemo 演示程序的屏幕截图。

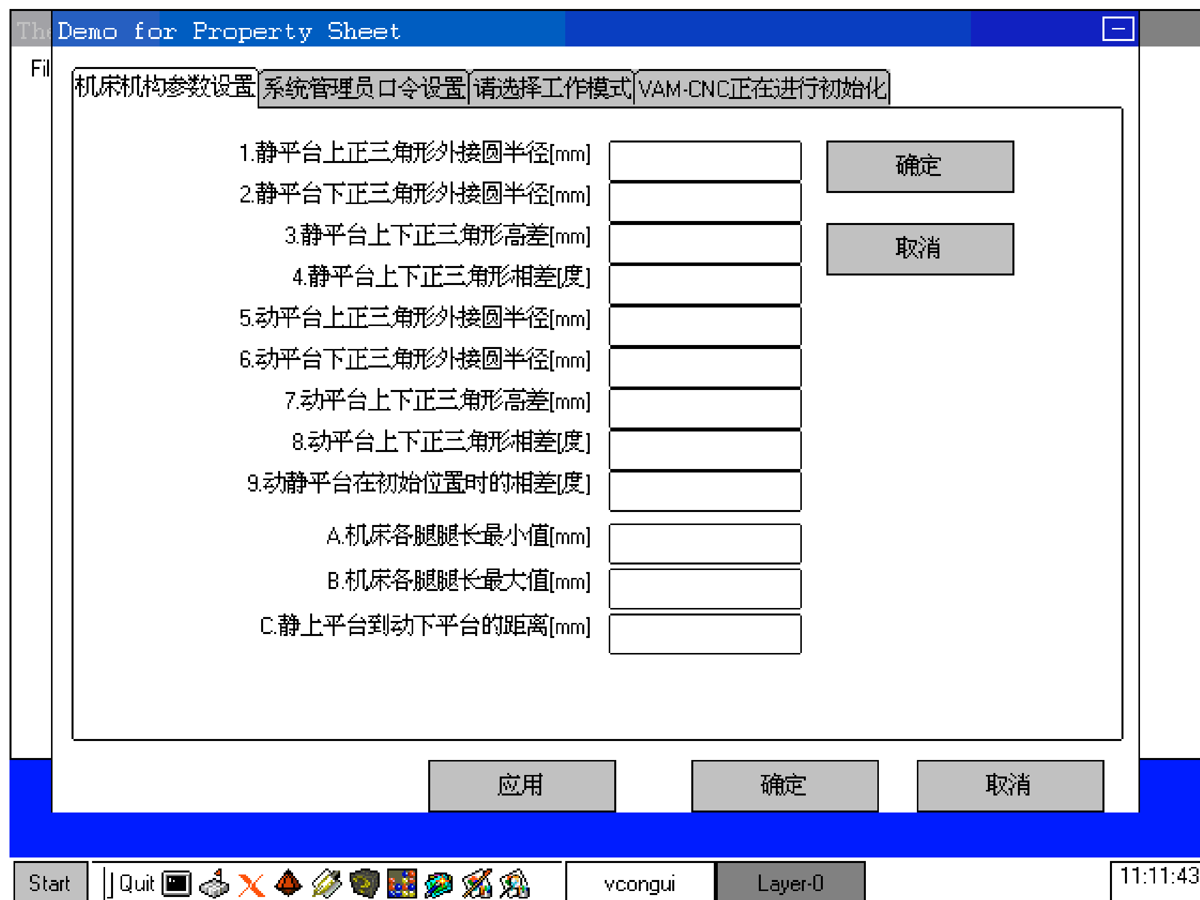


图 3-2 dlgdemo 程序

【注意】**MiniGUI-Lite** 采用的是客户/服务器架构。要运行 **MiniGUI-Lite** 应用程序，需要首先启动一个服务器程序。**MiniGUI-Lite** 的服务器是 **mginit**，只有启动了 **mginit**，才能启动并运行其他 **MiniGUI-Lite** 的应用程序。

4 MiniGUI 配置选项

从整体上讲，MiniGUI 的配置选项分两个阶段。第一阶段是在编译前确定的，第二阶段是在运行时确定的。本章将详细讲述这两个阶段的配置方法。

4.1 MiniGUI 的编译配置选项

假定用户使用的是 RedHat Linux 9 系统，下面是在 MiniGUI 的源代码目录下运行 ./configure --help 输出的结果（注意该命令在 Red Hat 低版本发行版上的输出可能有所不同）：

```
`configure' configures this package to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print `checking...' messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for `--cache-file=config.cache'
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or `..']

Installation directories:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [PREFIX]

By default, `make install' will install all the files in
`/usr/local/bin', `/usr/local/lib' etc.  You can specify
an installation prefix other than `/usr/local' using `--prefix',
for instance `--prefix=$HOME'.

For better control, use the options below.

Fine tuning of the installation directories:
  --bindir=DIR           user executables [EPREFIX/bin]
  --sbindir=DIR          system admin executables [EPREFIX/sbin]
  --libexecdir=DIR       program executables [EPREFIX/libexec]
  --datadir=DIR          read-only architecture-independent data [PREFIX/share]
  --sysconfdir=DIR       read-only single-machine data [PREFIX/etc]
  --sharedstatedir=DIR   modifiable architecture-independent data [PREFIX/com]
```

```
  --localstatedir=DIR    modifiable single-machine data [PREFIX/var]
  --libdir=DIR           object code libraries [EPREFIX/lib]
  --includedir=DIR       C header files [PREFIX/include]
  --oldincludedir=DIR    C header files for non-gcc [/usr/include]
  --infodir=DIR          info documentation [PREFIX/info]
  --mandir=DIR           man documentation [PREFIX/man]

Program names:
  --program-prefix=PREFIX            prepend PREFIX to installed program names
  --program-suffix=SUFFIX            append SUFFIX to installed program names
  --program-transform-name=PROGRAM   run sed PROGRAM on installed program names

System types:
  --build=BUILD     configure for building on BUILD [guessed]
  --host=HOST       cross-compile to build programs to run on HOST [BUILD]
  --target=TARGET   configure for building compilers for TARGET [HOST]

Optional Features:
  --disable-FEATURE       do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG]  include FEATURE [ARG=yes]
  --enable-shared=PKGS  build shared libraries default=yes
  --enable-static=PKGS  build static libraries default=yes
  --enable-fast-install=PKGS  optimize for fast installation default=yes
  --disable-libtool-lock  avoid locking (might break parallel builds)
  --disable-dependency-tracking Speeds up one-time builds
  --enable-dependency-tracking  Do not reject slow dependency extractors
  --enable-ownmalloc       use own implementation of malloc functions <default=no>
  --enable-ownstdio        use own implementation of stdio functions <default=no>
  --enable-lite            build MiniGUI-Lite version <default=no>
  --enable-standalone      build MiniGUI-Lite Stand-Alone version <default=no>
  --enable-incoreres       use incore resource instead file IO to initialize MiniGUI
<default=no>
  --enable-newgal          use NEWGAL and its engines <default=yes>
  --enable-fixedmath       use NEWGAL and its engines <default=yes>
  --enable-debug           build with debugging messages <default=no>
  --enable-tracemsg        trace messages of MiniGUI <default=no>
  --enable-msgstr          include symbol name of message <default=no>
  --enable-timerunitms     unit of timer is 10ms <default=yes>
  --enable-micemoveable    user can move window by using mouse <default=yes>
  --enable-dblclk          mouse button can do double click <default=yes>
  --enable-cursor          include cursor support (for MiniGUI-Lite) <default=yes>
  --enable-clipboard       include clipboard support <default=yes>
  --enable-svgalib         build the GAL and IAL engines on SVGALib (for old GAL and
MiniGUI-Threads) <default=no>
  --enable-libggi          build the GAL and IAL engines on LibGGI (for old GAL and M
iniGUI-Threads) <default=no>
  --enable-vga16gal        build the VGA 16-color mode GAL engine <default=no>
  --enable-nativegal       build the native FrameBuffer GAL engine <default=yes>
  --enable-galfbcon        support native FrameBuffer GAL engine on Linux FrameBuffer
 console <default=yes>
  --enable-galqvfb         support native FrameBuffer GAL engine on Qt Virtual FrameB
uffer <default=yes>
  --enable-galecoslcd      support native FrameBuffer GAL engine on eCose LCD interfa
ce <default=no>
  --enable-galucos2lcd     support native FrameBuffer GAL engine on uC/OS-II LCD inte
rface <default=no>
  --enable-coortrans_cw    support clockwise rotation of screen in the native FB GAL
engine <default=no>
```

```
  --enable-coortrans_ccw   support counterclockwise rotation of screen in the native
FB GAL engine <default=no>
  --enable-fblin1r         build the 1BPP FB subdriver of native graphics engine (MSB
 is right) <default=no>
  --enable-fblin1l         build the 1BPP FB subdriver of native graphics engine (MSB
 is left) <default=no>
  --enable-fblin2r         build the 2BPP FB subdriver of native graphics engine (MSB
 is right) <default=no>
  --enable-fblin2l         build the 2BPP FB subdriver of native graphics engine (MSB
 is left) <default=no>
  --enable-fblin4r         build the 4BPP FB subdriver of native graphics engine (MSB
 is right) <default=no>
  --enable-fblin4l         build the 4BPP FB subdriver of native graphics engine (MSB
 is left) <default=no>
  --enable-fblin8          build the 8BPP FB subdriver of native graphics engine <def
ault=yes>
  --enable-fblin16         build the 16BPP FB subdriver of native graphics engine <de
fault=yes>
  --enable-fblin24         build the 24BPP FB subdriver of native graphics engine (in
completed) <default=no>
  --enable-fblin32         build the 32BPP FB subdriver of native graphics engine <de
fault=no>
  --enable-fbvga16         build the VGA16 FB subdriver of native graphics engine (do
 not enable, dangrous) <default=no>
  --enable-ep7211ial       build the input engine for EP7211-based board <default=no>
  --enable-adsial          build the input engine for ADS Graphics Client board <defa
ult=no>
  --enable-ipaqial         build the input engine for iPAQ H3600 <default=no>
  --enable-l7200ial        build the input engine for L7200 <default=no>
  --enable-arm3000ial      build the input engine for ARM3000 <default=no>
  --enable-palm2ial        build the input engine for OKWAP Palm2 <default=no>
  --enable-mpc823ial       build the input engine for mpc823 <default=no>
  --enable-ucb1x00ial      build the input engine for ucb1x00 <default=no>
  --enable-px255bial       build the input engine for px255b <default=no>
  --enable-vr4181ial       build the input engine for NEC VR4181 debug board <default
=no>
  --enable-helioial        build the input engine for Helio Touch Panel <default=no>
  --enable-tfstbial        build the input engine for Tongfang STB <default=no>
  --enable-t800ial         build the input engine for MT T800 <default=no>
  --enable-mc68x328ial     build the input engine for uClinux touch screen palm/mc68e
z328 <default=no>
  --enable-smdk2410ial     build the input engine for SMDK2410 touch screen <default=
no>
  --enable-hh2410r3ial     build the input engine for HHARM2410R3 touch screen <defau
lt=no>
  --enable-dmgstbial       build the input engine for Intel DMG STB remote controller
 <default=no>
  --enable-fipsdfial       build the input engine for EM85xx Front Panel and Remote C
ontroller <default=no>
  --enable-svpxxial        build the input engine for WinBond SVPXX <Default=no>
  --enable-ads7846ial      build the input engine for Arca Tpanel Ads7846 <Default=no
>
  --enable-dummyial        build the Dummy IAL engine <default=yes>
  --enable-autoial         build the Automatic IAL engine <default=no>
  --enable-hhkbdirial      build the IAL engine for HHCF5249 PS/2 keyboard and IrDA <
default=no>
  --enable-ucos2ial        build the UCOS2 IAL engine <default=no>
  --enable-commial         build the COMM IAL engine <default=no>
```

```
  --enable-qvfbial          build the QVFB IAL engine <default=yes>
  --enable-wvfbial          build the WVFB IAL engine <default=no>
  --enable-utpmcial         build the IAL engine for UTPMC <default=no>
  --enable-nativeial        build the native (console) input engine <default=yes>
  --enable-nativeps2        build the native engine subdriver for PS2 mouse <default=y
es>
  --enable-nativeimps2      build the native engine subdriver for IntelligentMouse (IM
PS/2) mouse <default=yes>
  --enable-nativems         build the native engine subdirver for old MS serial mouse
<default=yes>
  --enable-nativems3        build the native engine subdirver for MS3 mouse <default=y
es>
  --enable-nativegpm        build the native engine subdirver for GPM daemon <default=
yes>
  --enable-textmode         Linux system have console (text mode) on FrameBuffer <defa
ult=yes>
  --enable-rbfsupport       include raw bitmap font support <default=yes>
  --enable-rbf24            include incore RBF font of ISO8859-1 12x24 (effective when
 use incore resource) <default=no>
  --enable-rbf16            include incore RBF font of ISO8859-1 8x16 (effective when
use incore resource) <default=no>
  --enable-rbfgb12          include incore RBF font of GB2312 12x12 (effective when us
e incore resource) <default=yes>
  --enable-rbfgb16          include incore RBF font of GB2312 16x16 (effective when us
e incore resource) <default=no>
  --enable-rbfgb24          include incore RBF font of GB2312 24x24 (effective when us
e incore resource) <default=no>
  --enable-vbfsupport       include var bitmap font support <default=yes>
  --enable-fontsserif       include incore font sansserif <default=yes>
  --enable-fontcourier      include incore font courier <default=yes>
  --enable-fontsymbol       include incore font symbol <default=no>
  --enable-fontvgas         include incore font VGAs <default=no>
  --enable-fonthelv         include incore font Helvetica <default=no>
  --enable-qpfsupport       build support for Qt Prerendered Font (QPF) <default=yes>
  --enable-ttfsupport       build support for TrueType font <default=yes>
  --enable-type1support     build support for Adobe Type1 font <default=yes>
  --enable-latin2support    include East European (Latin 2, ISO-8859-2) charset suppor
t <default=no>
  --enable-latin3support    include South European (Latin 3, ISO-8859-3) charset suppo
rt <default=no>
  --enable-latin4support    include North European (Latin 4, ISO-8859-4) charset suppo
rt <default=no>
  --enable-cyrillicsupport include Cyrillic (ISO-8859-5) charset support <default=no>
  --enable-arabicsupport    include Arabic (ISO-8859-6) charset support <default=no>
  --enable-greeksupport     include Greek (ISO-8859-7) charset support <default=no>
  --enable-hebrewsupport    include Hebrew (ISO-8859-8) charset support <default=no>
  --enable-latin5support    include Turkish (Latin 5, ISO-8859-9) charset support <def
ault=no>
  --enable-latin6support    include Nordic, Latin 6, ISO-8859-10) charset support <def
ault=no>
  --enable-thaisupport      include Thai (ISO-8859-11) charset support <default=no>
  --enable-latin7support    include Latin 7 (ISO-8859-13) charset support <default=no>
  --enable-latin8support    include Latin 8 (ISO-8859-14) charset support <default=no>
  --enable-latin9support    include Latin 9 (ISO-8859-15, West Extended) charset suppo
rt <default=yes>
  --enable-latin10support   include Latin 10 (ISO-8859-16, Romanian) charset support <
default=no>
  --enable-gbsupport        include EUC encoding of GB2312 charset support <default=ye
```

```
s>
  --enable-gbksupport      include GBK charset support <default=yes>
  --enable-gb18030support  include GB18030-0 charset support <default=no>
  --enable-big5support     include BIG5 charset support <default=yes>
  --enable-euckrsupport    include support for EUC encoding of KSC5636 and KSC5601 ch
arsets <default=no>
  --enable-eucjpsupport    include support for EUC encoding of JISX0201 and JISX0208
charsets <default=no>
  --enable-shiftjissupport include support for Shift-JIS encoding of JISX0201 and JIS
X0208 charsets <default=no>
  --enable-unicodesupport  include UNICODE (ISO-10646-1 and UTF-8 encoding) support <
default=yes>
  --enable-kbdfrpc         include keyboard layout for French PC keyboard (non-US 102
 keys) <default=no>
  --enable-kbdfr           include keyboard layout for French <default=no>
  --enable-kbdde           include keyboard layout for German <default=no>
  --enable-kbddelatin1     include keyboard layout for German Latin1 <default=no>
  --enable-kbdit           include keyboard layout for Italian <default=no>
  --enable-kbdes           include keyboard layout for Spanish <default=no>
  --enable-kbdescp850      include keyboard layout for Spanish CP850 <default=no>
  --enable-savebitmap      include SaveBitmap-related functions <default=yes>
  --enable-pcxsupport      include PCX file support <default=no>
  --enable-lbmsupport      include LBM/PBM file support <default=no>
  --enable-tgasupport      include TGA file support <default=no>
  --enable-gifsupport      include GIF file support <default=yes>
  --enable-jpgsupport      include JPG file support <default=yes>
  --enable-pngsupport      include PNG file support <default=yes>
  --enable-imegb2312       include IME (GB2312) support <default=yes>
  --enable-imegb2312py     include IME (GB2312) Intelligent Pinyin module <default=ye
s>
  --enable-aboutdlg        include About Dialog Box <default=yes>
  --enable-savescreen      include code for screenshots <default=yes>
  --enable-grayscreen      target is a gray screen <default=no>
  --enable-tinyscreen      target is a tiny-size screen <default=no>
  --enable-ctrlstatic      include STATIC control <default=yes>
  --enable-ctrlbutton      include BUTTON control <default=yes>
  --enable-ctrlsimedit     include Simple EDIT control <default=yes>
  --enable-ctrlsledit      include Single-Line EDIT control <default=yes>
  --enable-ctrloldmedit    include old Multi-Line EDIT control <default=no>
  --enable-ctrllistbox     include LISTBOX control <default=yes>
  --enable-ctrlpgbar       include PROGRESSBAR control <default=yes>
  --enable-ctrltoolbar     include TOOLBAR control <default=yes>
  --enable-ctrlnewtoolbar  include NEWTOOLBAR control <default=yes>
  --enable-ctrlmenubtn     include MENUBUTTON control <default=yes>
  --enable-ctrltrackbar    include TRACKBAR control <default=yes>
  --enable-ctrlcombobox    include COMBOBOX control <default=yes>
  --enable-ctrlpropsheet   include PROPSHEET control <default=yes>
  --enable-ctrlscrollview  include SCROLLVIEW and SCROLLWINDOW controls <default=yes>
  --enable-ctrltextedit    include TEXTEDIT control which is based-on SCROLLVIEW cont
rol <default=yes>
  --enable-extctrlmonthcal     include MONTHCALENDAR control in MiniGUIExt library <d
efault=yes>
  --enable-extctrltreeview     include TREEVIEW control in MiniGUIExt library <defaul
t=yes>
  --enable-extctrlspinbox      include SPINBOX control in MiniGUIExt library <default
=yes>
  --enable-extctrlcoolbar      include COOLBAR control in MiniGUIExt library <default
=yes>
```

```
  --enable-extctrllistview     include LISTVIEW control in MiniGUIExt library <defaul
t=yes>
  --enable-extctrlgrid         include GRID control in MiniGUIExt library <default=ye
s>
  --enable-extctrlanimation    include ANIMATION control and GIF87a/GIF89a support in
 MiniGUIExt library <default=yes>
  --enable-extskin             include skin support in MiniGUIExt library <default=ye
s>
  --enable-libvcongui          include support for Virtual Console on Linux <default=
yes>
  --enable-videodummy          include dummy NEWGAL engine <default=yes>
  --enable-videofbcon          include FrameBuffer console NEWGAL engine <default=yes
>
  --enable-videoqvfb           include Qt Virtual FrameBuffer NEWGAL engine <default=
yes>
  --enable-videowvfb           include windows Virtual FrameBuffer NEWGAL engine <def
ault=no>
  --enable-videocommlcd        include NEWGAL engine for Common LCD <default=no>
  --enable-videoecoslcd        include NEWGAL engine for ECOS LCD <default=yes>
  --enable-videoucos2lcd       include NEWGAL engine for UC/OS-II LCD <default=yes>
  --enable-videoem85xxosd      include NEWGAL engine for EM85xx OSD <default=yes>
  --enable-videosvpxxosd       include SVPXXOSD NEWGAL engine <default=no>
  --enable-videobf533          include NEWGAL engine for BF533 OSD via SPI <default=n
o>
  --enable-videoutpmc          include NEWGAL engine for UTPMC <default=no>
  --enable-adv2dapi            include advanced 2D graphics APIs <default=yes>

Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-gnu-ld           assume the C compiler uses GNU ld default=no
  --with-pic              try to use only PIC/non-PIC objects default=use both
  --with-osname=linux/uclinux/ecos/ucos2/vxworks/swlinux
  --with-style=pc3d/flat/phone

Some influential environment variables:
  CC          C compiler command
  CFLAGS      C compiler flags
  LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
              nonstandard directory <lib dir>
  CPPFLAGS    C/C++ preprocessor flags, e.g. -I<include dir> if you have
              headers in a nonstandard directory <include dir>
  CPP         C preprocessor

Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.
```

上面这些参数是我们在 configure 脚本中设置好的命令行参数，可以控制在要编译的 MiniGUI 中包含支持哪些功能的代码。比如，运行：

```
$ ./configure --with-style=phone  --enable-lite --enable-standalone
```

就可以将 MiniGUI 配置成具有 phone 风格的外观，且具有 MiniGUI-Standalone 运行模式。相反，如果运行：

```
$ ./configure --with-style=flat  --enable-lite
```

将 MiniGUI 配置成 MiniGUI-Lite 模式，使用 flat 风格。不带任何参数执行 ./configure 命令将按照默认选项生成 Makefile。注意在每个选项的说明中都给出了默认设置：default=yes 或者 default=no。

4.1.1 通用选项

下面这些选项是 GNU 风格软件包使用的一些重要的通用选项：

- --prefix=PREFIX

该选项用于指定 MiniGUI 函数库的安装路径。默认的安装路径是 /usr/local。如果运行：

```
./configure --prefix=/home/test
```

那么在执行 make install 之后，函数库、头文件以及参考手册页将被安装在 /home/test/lib、/home/test/include、/home/test/man 目录下。

该选项和 --build、--host 以及 --target 等选项对 MiniGUI 和应用程序的交叉编译非常重要。本手册第 5 章将详细讲述这些选项在交叉编译时的使用。

- --enable-static 和 --enable-shared

这两个选项指定生成函数库的静态库和动态库版本。如果不需要生成静态库，则可以使用 --disable-static 选项，这样，将缩短编译函数库的时间。

4.1.2 MiniGUI 选项

接下来一些具体的配置 MiniGUI 的选项基本上都是基于 --disable-FEATURE 和 --enable-FEATURE 实现的。--disable-FEATURE 选项禁止某项特性，也就是在函数库中将不对该特性进行支持。--enable-FEATURE 选项打开某项特性，也就是在函数库中将对该特性进行支持。

另外，MiniGUI 配置脚本还提供了 --with 选项，该选项可用来从多个可选项中指定其中某一个选项。MiniGUI 所运行的操作系统及控件风格通过 with 选项指定。

- --with-osname=linux/uclinux/ecos/ucos2/vxworks/swlinux

用于指定 MiniGUI 所运行的操作系统，默认为 linux。如果要让 MiniGUI 在 uClinux 上运行，应使用 --with-osname=uclinux 选项。

■ --with-style=pc3d/flat/phone

用于指定 MiniGUI 控件的外观风格，默认为 pc3d，即类似 PC 的三维风格。其它两个选项分别是：flat：适合灰度屏的平板风格；phone：适合手持终端的华丽风格。

其它的 MiniGUI 配置选项，均是开关选项（下面的描述只给出选项本身的名称）：

■ lite [default=no]

打开该选项将生成基于进程的 MiniGUI-Lite 运行模式。目前 MiniGUI 函数库有三种运行模式：Threads、Lite 和 Standalone；默认生成 Threads 模式。这三种运行模式之间具有较大的区别，应用范围也有所不同；它们之间的差异在附录 A 中给出。

■ standalone [default=no]

打开上面的 lite 选项以及本选项之后，将生成基于进程且单独运行的 MiniGUI-Standalone 运行模式。

■ incoreres [default=no]

打开该选项将使用 MiniGUI 的内建资源。MiniGUI 运行时所需要的字体、位图等，都会被编译到 libminigui 库中，包括的运行时配置信息在内。

■ micemoveable [default=yes]

关闭该选项将禁止用户使用鼠标来移动窗口。在很多的嵌入式系统上，不使用层叠式的多窗口用户界面，也不需要移动窗口，这时，就可以使用 --disable-micemoveable 配置选项。

■ newgal [default=yes]

关闭该选项将不使用 NEWGAL 和 NEWGDI 接口，而使用老的 GAL 和 GDI 接口。默认情况下，函数库使用 NEWGAL 和 NEWGDI 接口。NEWGAL 引擎只支持 8 位色以上的线性显示模式，但功能强大，但不支持低于 8 位色的那些模式（比如 16 色及低于 16 色的灰度显示模式等等）。如果需要支持低于 8 位色的那些模式，则应该使用老的 GAL 引擎，它最低可以支持 1 级灰度（即黑白屏幕）。如果用户需要使用老的 GAL 接口来支持较低的颜色深度，则应该使用如下选项：

```
--disable-newgal
```

■ ep7211ial 等

ep7211ial、adsial、ipaqial、mpc823ial、ucb1x00ial、px255bial、vr4181ial、helioial、t800ial、smdk2410ial、dmgstbial、fipial 等选项是针对特定开发板的 IAL 引擎选项，这些开发板分别是：EP7211、ADS Graphics Client（基于 StrongARM SA1110）、Motorola PowerPC 823 开发板（鑫富瑞普）、学生 PC（中基软）、Intel xScale 开发板、VR4181、Helio、T800、S3C2401 开发板（远峰）、Intel DMG 机顶盒、Sigma Designs RealMagic EM85xx 机顶盒等。如果您的嵌入式开发板不在上述之列，则需要根据自己的目标板开发驱动程序及 IAL 引擎。不过，您也可以使用 MiniGUI 内建的 DUMMY IAL 引擎（--enable-dummyial）来暂时跳过对输入的处理。

■ nativegal [default=yes]

该选项用来控制 OLDGAL 接口的 NATIVE/FBCON 引擎。使用该选项将在库中包含老 GAL 接口的 NATIVE/FBCON 引擎，这个引擎是建立在 FrameBuffer 基础上的，而且可以支持各种不同的颜色深度。在最终的引擎中包含对哪些颜色深度的支持，取决于 --enable-fblinXX 选项（见下）。

■ coortrans_cw [default=no]

打开该选项将激活老 GAL 接口中的坐标转换功能，这个功能将使屏幕的显示以顺时针旋转 90 度，此功能通常在 COMPAQ iPAQ H3600 系列的产品中使用。

■ coortrans_ccw [default=no]

打开该选项将激活老 GAL 接口中的坐标转换功能，这个功能将使屏幕的显示以逆时针旋转 90 度，此功能通常在 COMPAQ iPAQ H3800 系列的产品中使用。

■ fblinXX

fblin1、fblin2、fblin_2、fblin4、fblin8、fblin16、fblin24、fblin32 等选项用来指定老 GAL 接口的 NATIVE 引擎中包含对哪些颜色深度的支持。举例来讲，如果要支持单色 LCD，则应该使用如下选项：

```
--disable-newgal  --enable-nativegal --enable-fblin1
```

■ vga16gal [default=no]

打开该选项将包含支持标准 VGA 16 色模式的图形引擎。vga16 是 OLDGAL 接口的引擎。如果用户想要使用 PC 的标准 VGA 16 色图形系统，则需要使用如下选项：

```
--disable-newgal  --enable-vga16gal
```

■ nativeial [default=yes]

该选项用来指定是否包含标准 PC 控制台上的 IAL 引擎，可用来支持 PC 上的键盘及常见鼠标。在 MiniGUI.cfg 中指定 ial_engine=console 可指定 MiniGUI 使用该引擎。

■ nativeps2 等

该选项以及 nativeimps2、nativems、nativems3、nativegpm 等选项，用来指定是否在 MiniGUI 库中包含对特定鼠标协议的支持，分别是 PS2、IMPS2、MS（老式串口鼠标）、MS3（带中键的串口鼠标）以及 GPM。通过 GPM 协议以及 Linux 上的 GPM 守护进程，MiniGUI 可以支持几乎所有类型的鼠标。有关 GPM 守护进程的使用，请参阅附录 B。

■ textmode [default=yes]

关闭该选项可用于无字符模式的 Linux 系统（即没有字符控制台的 Linux 系统，对某些嵌入式设备有用）。

■ cursor [default=yes]

如果您的目标系统没有鼠标、触摸屏等定点设备，则不需要显示鼠标光标，这时，就可以关闭该选项从而取消对鼠标光标的支持。

■ rbfsupport [default=yes]

关闭该选项可禁止对 Raw Bitmap Font（RBF）的支持。不过，MiniGUI 的正常运行需要装载至少一种 Raw Bitmap 字体，因此，不建议使用该选项禁止对 RBF 字体的支持。

■ rbfgb12 等

该选项以及 rbfgb12、rbfgb16、rbfgb24 等选项，可控制是否在 libminigui 库中内建 GB2312 的 12、16 及 24 点阵字体。这些选项仅在 incoreres 选项被打开时有效。

■ vbfsupport [default=yes]

关闭该选项可禁止 MiniGUI 对 Var Bitmap Font（VBF）的支持，同时将禁止在库中包含内建的 VBF 字体。MiniGUI 启动时，也将忽略 MiniGUI.cfg 文件中的 [varbitmapfonts] 段。

■ fontsserif 等

该选项以及 fontcourier、fontsymbol、fonthelv、fontvgas、fonthelv 选项，可禁止内建在 libminigui 库中的 VBF 字体，包括 SanSerif、Courier、Symbol、Helvetica 以

及若干 VGA 字体。

■ qpfsupport [default=yes]

使用该选项将在库中包含对 Qt/Embedded Prerendered Font（QPF）字体的支持。因为 QPF 字体使用 UNICODE 编码，因此，允许对 QPF 字体的支持，将自动打开 MiniGUI 的 UNICODE 支持。MiniGUI 在启动时，将读取 MiniGUI.cfg 中 [qpf] 段的定义，并装载该段中定义的 QPF 字体。

■ ttfsupport [default=yes]

使用该选项将在库中包含对 TrueType 字体的支持。MiniGUI 使用 FreeType 版本 1.3.0 来提供对 TrueType 字体的渲染。如果您的系统中没有 FreeType 1.3.0，配置脚本将自动禁止该选项。需要注意的是，FreeType 2 的接口和 FreeType 1 不兼容，所以，MiniGUI 不能在只有 FreeType 2 的系统上正常编译，这时，您需要使用 --disable-ttfsupport 手动禁止对 TrueType 字体的支持。

■ type1support [default=yes]

使用该选项将在库中包含对 Adobe Type1 字体的支持。MiniGUI 使用 t1 库来提供对 Type1 字体的渲染。如果您的系统中没有安装 t1 库，则配置脚本会自动禁止该选项。

■ latin2support 等

latin2support、latin3support、cyrillicsupport、arabicsupport、greeksupport、hebrewsupport、latin5support、latin6support、thaisupport、latin7support、latin8support、latin9support、latin10support 等选项，用来控制对 ISO8859-2 到 ISO8859-16 字符集的支持。这些字符集均是单字节字符集。

■ gbsupport 等

gbsupport、gbksupport、gb18030support、big5support、euckrsupport、eucjpsupport、shiftjissupport、unicodesupport 等选项，分别用来控制对 GB2312、GBK、GB18030、BIG5、EUCKR、EUCJP、SHIFTJIS、UNICODE 等多字节字符集/编码系统的支持。

前面提到，在打开对 QPF 及 TTF 字体的支持后，将自动包含对 UNICODE 的支持，同时还将包含已支持各字符集到 UNICODE UCS2 编码的映射表。

■ kbdfrpc 等

kbdfrpc、kbdfr、kbdde、kbddelatin1、kbdit、kbdes、kbdescp850 等选项用来指定 MiniGUI 包含哪些键盘布局，一般无需设置。默认不包含上述键盘布局。

■ savebitmap [default=yes]

控制是否提供将 BITMAP 结构保存为 Windows BMP 文件格式的支持。

■ pcxsupport 等

pcxsupport、lbmsupport、tgasupport、gifsupport、jpgsupport、pngsupport 等选项分别用来控制是否在库中包含对 PCX、LBM、TGA、GIF、JPG、PNG 图片格式的支持。注意对 JPG 和 PNG 图片格式的支持，需要系统中安装有对应版本的 libjpeg 和 libpng 库。

■ imegb2312 [default=yes]

用来控制是否包含 GB2312 中文简体输入法。

■ imegb2312pinyin [default=yes]

用来控制是否包含 GB2312 智能拼音输入法。

■ aboutdlg [default=yes]

用来控制是否包含 About MiniGUI 对话框。

■ savescreen [default=yes]

用来控制是否响应 <PrntScrn> 键并保存屏幕到当前目录。在将 MiniGUI 移植到嵌入式系统中时，最好采用该选项禁止屏幕的保存功能。包含该功能时，用户按 <PrntScrn> 键将在当前目录保存运行时刻的屏幕图，按 <Ctrl+PrntScrn> 键将保存当前活动窗口的屏幕图。

■ grayscreen [default=no]

该选项用来控制某些屏幕元素的绘制方式。如果您的目标平台采用灰度屏幕，则建议打开该选项。

■ tinyscreen [default=no]

该选项用来控制某些屏幕元素（比如 MessageBox）的绘制方式。如果您的目标平台具有很小的屏幕，比如小于 320x240，则建议打开该选项。

■ ctrlstatic 等

ctrlstatic、ctrlbutton、ctrlsimedit、ctrlsledit、ctrloldmledit、ctrllistbox、ctrlpgbar、ctrltoolbar、ctrlnewtoolbar、ctrlmenubtn、ctrltrackbar、ctrlcombobox、ctrlpropsheet、ctrlscrollview、ctrltextedit 等选项分别用来控制是否在库中包含静态框、按钮、简单编辑框（只能处理等宽字体和 GB2312 字符集）、单行编辑框、老的多行编辑框、列表框、进度条、工具栏、新工具栏、菜单按钮、跟踪条、组合框、属性页、滚动视图、增强文本编辑框等控件。

■ extctrlmonthcal 等

extctrlmonthcal、extctrltreeview、extctrlspinbox、extctrlcoolbar、extctrllistview、extctrlgrid、extctrlanimation 等选项用来控制是否在 MiniGUI Ext 库中包含月历、树型、SpinBox、酷工具栏、列表型、网格及动画等控件。

■ extskin [default=yes]

该选项用来控制是否在 MiniGUI Ext 库中包含对皮肤界面（SKIN）的支持。

■ libvcongui [default=yes]

该选项用来控制是否编译 libvcongui 库，该 libvcongui 提供的功能仅限于 Linux 操作系统。

■ adv2dapi [default=yes]

该选项用来控制是否包含高级二维绘图函数接口。这些接口在 NEWGAL 之上提供，并且需要数学库的支持。

■ video-dummy [default=yes]

该选项控制是否打开“哑”的 NEWGAL 引擎，也就是不作实际输出的图形引擎。

■ video-fbcon [default=yes]

该选项控制是否打开控制台上 FrameBuffer 的支持，它是 NEWGAL 接口的默认图形引擎。

■ video-qvfb [default=yes]

该选项控制是否打开对 QT 虚拟 FrameBuffer 机制的支持。利用 qvfb 支持，可以在 X Window 的窗口中运行 MiniGUI 程序，这样可大大方便应用程序的调试。注意，qvfb 是 NEWGAL 接口的一个引擎，选择该选项将在 libminigui 中包含对 qvfb 的支持。在编译安装函数库之后，如果要使用 qvfb 引擎，则需要：

1) 启动 qvfb 程序。
2) 设置 qvfb 的分辨率和颜色深度。
3) 修改 MiniGUI.cfg 文件，确保指定 gal_engine 为 qvfb，并在 [qvfb] 段中指定对应的显示模式。

■ videoecoslcd [default=no]

该选项控制是否打开对 eCos 操作系统上 LCD 接口的支持。

■ videoem85xxosd

该选项控制是否打开对 RealMagic EM85xx OSD 设备的支持。

4.1.3 内建式资源选项

如前所述，当打开 incoreres 选项时，MiniGUI 将使用内建在函数库中的资源。默认情况下系统使用的位图、图标和字体等资源从文件中载入，如果使用内建式资源的话，程序不需要 MiniGUI 系统资源文件和配置文件也可运行。和内建式资源相关的选项有：

■ rbfgb12、rbfgb16、rbfgb24：内建式 GB2312 12 点阵、16 点阵及 24 点阵 RBF 字体支持。只有在 enable-incoreres 时才有效。

■ fontsserif：内建式 sansserif 字体支持。

■ fontcourier：内建式 courier 字体支持。

■ fontsymbol：内建式 symbol 字体支持。

■ fontvgas：内建式 VGA 字体支持，通常都是等宽字体。

以上就是 MiniGUI 1.6.x 版本所支持的绝大多数配置选项，用户可以根据自己的需求进行调整。

4.1.4 支持中文显示的最小 MiniGUI 函数库配置选项

在 MiniGUI 1.6.x 版本中包含了一个 buildlib-min 脚本。该脚本内容如下：

```
#!/bin/sh

./configure \
    --with-style=flat \
    --enable-grayscreen \
    --enable-tinyscreen \
    --disable-micemoveable \
    --disable-newgal \
    --disable-fblin32 \
    --disable-cursor \
    --disable-vbfsupport \
    --disable-qpfsupport \
    --disable-ttfsupport \
    --disable-type1support \
    --disable-latin9support \
    --disable-gbksupport \
    --disable-big5support \
    --disable-unicodesupport \
    --disable-savebitmap \
    --disable-jpgsupport \
    --disable-pngsupport \
    --disable-imegb2312 \
    --disable-imegb2312py \
    --disable-aboutdlg \
    --disable-savescreen
```

利用这个脚本可将 MiniGUI 配置成支持中文显示的最小函数库，从而使生成的 MiniGUI 函数库较小。由这个脚本配置的 MiniGUI 函数库包含或不包含如下功能：

- 将 MiniGUI 编译为 MiniGUI-Threads。
- 使用老的 GAL 接口，并包含 NATIVE 引擎，支持 8 位色 和 16 位色显示模式。
- 不支持 VBF 字体。
- 不支持 TTF 字体。
- 不支持 Type1 字体。
- 不包含对 Latin2 等字符集的支持，只包含对 Latin1，即 ISO8859-1 字符集的支持。
- 不包含 GBK 字符集、BIG5 字符集的支持，只包含对 GB2312 字符集的支持（对不需要中文支持的系统，甚至可以利用 --disable-gb2312support 将 GB2312 字符集的支持取消）。
- 不包含 GB2312 输入法支持。
- 不包含 BITMAP 保存支持。
- 不包含对 JPEG、PNG 图片格式的支持。
- 不包含“AboutMiniGUI”对话框。

■ 不包含保存屏幕的功能。

在上述配置基础上，用户还可以根据需求取消一些功能。

4.2 MiniGUI 的运行时配置文件：MiniGUI.cfg

下面是 1.6.x 版本默认安装的 MiniGUI 函数库的运行环境配置文件的内容：

```
# MiniGUI Ver 1.6.x.
# This configuration file is for 3D window style.
#
# Copyright (C) 2002~2004 Feynman Software
# Copyright (C) 1998~2002 Wei Yongming.
#
# Web:   http://www.minigui.com
# Web:   http://www.minigui.org
#
# This configuration file must be installed in /etc,
# /usr/local/etc or your home directory. When you install it in your
# home directory, it should be named ".MiniGUI.cfg".
#
# The priority of above configruation files is ~/.MiniGUI.cfg,
# /usr/local/etc/MiniGUI.cfg, and then /etc/MiniGUI.cfg.
#
# If you change the install path of MiniGUI resource, you should
# modify this file to meet your configuration.
#
# NOTE:
# The format of this configuration file has changed since the last release.
# Please DONT forget to provide the latest MiniGUI.cfg file for your MiniGUI.
#

[system]
# GAL engine
gal_engine=fbcon

# IAL engine
ial_engine=console

mdev=/dev/mouse
mtype=IMPS2

[fbcon]
defaultmode=1024x768-16bpp

[qvfb]
defaultmode=640x480-16bpp
display=0

# The first system font must be a logical font using RBF device font.
[systemfont]
font_number=6
font0=rbf-fixed-rrncnn-8-16-ISO8859-1
font1=*-fixed-rrncnn-*-16-GB2312
font2=*-Courier-rrncnn-*-16-GB2312
font3=*-SansSerif-rrncnn-*-16-GB2312
```

```
font4=*-Times-rrncnn-*-16-GB2312
font5=*-Helvetica-rrncnn-*-16-GB2312

default=0
wchar_def=1
fixed=1
caption=2
menu=3
control=3

[rawbitmapfonts]
font_number=4
name0=rbf-fixed-rrncnn-8-16-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/8x16-iso8859-1.bin
name1=rbf-fixed-rrncnn-16-16-GB2312.1980-0
fontfile1=/usr/local/lib/minigui/res/font/song-16-gb2312.bin
name2=rbf-fixed-rrncnn-6-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/6x12-iso8859-1.bin
name3=rbf-fixed-rrncnn-12-12-GB2312.1980-0
fontfile3=/usr/local/lib/minigui/res/font/song-12-gb2312.bin

[varbitmapfonts]
font_number=6
name0=vbf-Courier-rrncnn-8-13-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/Courier-rr-8-13.vbf
name1=vbf-Helvetica-rrncnn-11-12-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/Helvetica-rr-11-12.vbf
name2=vbf-Times-rrncnn-10-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/Times-rr-10-12.vbf
name3=vbf-Courier-rrncnn-10-15-ISO8859-1
fontfile3=/usr/local/lib/minigui/res/font/Courier-rr-10-15.vbf
name4=vbf-Helvetica-rrncnn-15-16-ISO8859-1
fontfile4=/usr/local/lib/minigui/res/font/Helvetica-rr-15-16.vbf
name5=vbf-Times-rrncnn-13-15-ISO8859-1
fontfile5=/usr/local/lib/minigui/res/font/Times-rr-13-15.vbf

[qpf]
font_number=3
name0=qpf-unifont-rrncnn-16-16-ISO8859-1,ISO8859-15,GB2312,GBK,BIG5
fontfile0=/usr/local/lib/minigui/res/font/unifont_160_50.qpf
name1=qpf-times-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile1=/usr/local/lib/minigui/res/font/smoothtimes_100_50.qpf
name2=qpf-helvetica-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile2=/usr/local/lib/minigui/res/font/helvetica_100_50.qpf
name3=qpf-micro-rrncnn-4-4-ISO8859-1,ISO8859-15
fontfile3=/usr/local/lib/minigui/res/font/micro_40_50.qpf

[truetypefonts]
font_number=3
name0=ttf-arial-rrncnn-0-0-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/arial.ttf
name1=ttf-times-rrncnn-0-0-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/times.ttf
name2=ttf-pinball-rrncnn-0-0-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/pinball.ttf

[type1fonts]
font_number=0
```

```
name0=type1-Charter-rrncnn-0-0-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/bchr.pfb
name1=type1-Charter-rincnn-0-0-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/bchri.pfb
name2=type1-Charter-brncnn-0-0-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/bchb.pfb
name3=type1-Charter-bincnn-0-0-ISO8859-1
fontfile3=/usr/local/lib/minigui/res/font/bchbi.pfb
name4=type1-Courier-rrncnn-0-0-ISO8859-1
fontfile4=/usr/local/lib/minigui/res/font/dcr10.pfb
name5=type1-Courier-rincnn-0-0-ISO8859-1
fontfile5=/usr/local/lib/minigui/res/font/dcti10.pfb
name6=type1-Courier-brncnn-0-0-ISO8859-1
fontfile6=/usr/local/lib/minigui/res/font/dcbx10.pfb
name7=type1-Courier-bincnn-0-0-ISO8859-1
fontfile7=/usr/local/lib/minigui/res/font/dcbxti10.pfb
name8=type1-eufm10-rrncnn-0-0-ISO8859-1
fontfile8=/usr/local/lib/minigui/res/font/eufm10.pfb

[mouse]
dblclicktime=300

[event]
timeoutusec=300000
repeatusec=50000

[cursorinfo]
# Edit following line to specify cursor files path
cursorpath=/usr/local/lib/minigui/res/cursor/
cursornumber=23
cursor0=d_arrow.cur
cursor1=d_beam.cur
cursor2=d_pencil.cur
cursor3=d_cross.cur
cursor4=d_move.cur
cursor5=d_sizenesw.cur
cursor6=d_sizens.cur
cursor7=d_sizenwse.cur
cursor8=d_sizewe.cur
cursor9=d_uparrow.cur
cursor10=d_none.cur
cursor11=d_help.cur
cursor12=d_busy.cur
cursor13=d_wait.cur
cursor14=g_rarrow.cur
cursor15=g_col.cur
cursor16=g_row.cur
cursor17=g_drag.cur
cursor18=g_nodrop.cur
cursor19=h_point.cur
cursor20=h_select.cur
cursor21=ho_split.cur
cursor22=ve_split.cur

[iconinfo]
# Edit following line to specify icon files path
iconpath=/usr/local/lib/minigui/res/icon/
# Note that max number defined in source code is 5.
```

```
iconnumber=5
icon0=form.ico
icon1=w95mbx01.ico
icon2=w95mbx02.ico
icon3=w95mbx03.ico
icon4=w95mbx04.ico

# default icons for TREEVIEW control
fold=fold.ico
unfold=unfold.ico

# default icons for new OpenFileDialogBox
dir=folder.ico
file=textfile.ico

[bitmapinfo]
# Edit following line to specify bitmap files path
bitmappath=/usr/local/lib/minigui/res/bmp/
# Note that max number defined in source code is 7
bitmapnumber=2
bitmap0=capbtns.bmp
bitmap1=arrows.bmp
# use large bitmap if your default font is 16 pixel height.
# bitmap1=arrows16.bmp
bitmap2=none
bitmap3=none
bitmap4=none
bitmap5=none

# background picture, use your favirate photo
bitmap6=none

# bitmap used by BUTTON control
button=button.bmp
pushbutton=none
pushedbutton=none

# bitmap used by LISTBOX control
checkmark=checkmark.bmp

# bitmap used by COMBOBOX control
downarrow=downarrow.bmp
updownarrow=updownarrow.bmp
leftrightarrow=leftrightarrow.bmp

# bitmap used by SPINBOX control
spinbox_vert=spinbox-vert.bmp
spinbox_horz=spinbox-horz.bmp

# bitmap used by listview control
lvfold=lvfold.bmp
lvunfold=lvunfold.bmp

# bitmap used by IME window
IMEctrlbtn=shurufa.bmp

# bitmap used by About dialog box
logo=MiniGUI256.bmp
```

```
# logo=MiniGUI16.bmp

[bgpicture]
position=center
# position=upleft
# position=downleft
# position=upright
# position=downright
# position=upcenter
# position=downcenter
# position=vcenterleft
# position=vcenterright
# position=none

[mainwinmetrics]
minwidth=50
minheight=50
border=2
thickframe=2
thinframe=1
captiony=+4
iconx=16
icony=16
menubary=+0
menubaroffx=8
menubaroffy=5
menuitemy=+0
intermenuitemx=12
intermenuitemy=2
menuitemoffx=18
menutopmargin=4
menubottommargin=4
menuleftmargin=4
menurightmargin=4
menuitemminx=64
menuseparatory=4
menuseparatorx=4
sb_height=14
sb_width=16
sb_interx=2
cxvscroll=12
cyvscroll=12
cxhscroll=12
cyhscroll=12
minbarlen=9
defbarlen=18

[windowelementcolors]
bkc_caption_normal=0x00808080
fgc_caption_normal=0x00C0C0C0
bkc_caption_actived=0x00800000
fgc_caption_actived=0x00FFFFFF
bkc_caption_disabled=0x00808080
fgc_caption_disabled=0x00C0C0C0
wec_frame_normal=0x00000000
wec_frame_actived=0x00FF0000
wec_frame_disabled=0x00000000
bkc_menubar_normal=0x00C0C0C0
```

```
fgc_menubar_normal=0x00000000
bkc_menubar_hilite=0x00800000
fgc_menubar_hilite=0x00FFFFFF
fgc_menubar_disabled=0x00808080
bkc_menuitem_normal=0x00C0C0C0
fgc_menuitem_normal=0x00000000
bkc_menuitem_hilite=0x00800000
fgc_menuitem_hilite=0x00FFFFFF
fgc_menuitem_disabled=0x00808080
bkc_pppmenutitle=0x00C0C0C0
fgc_pppmenutitle=0x00FF0000
fgc_menuitem_frame=0x00C66931
wec_3dbox_normal=0x00C0C0C0
wec_3dbox_reverse=0x00000000
wec_3dbox_light=0x00FFFFFF
wec_3dbox_dark=0x00808080
wec_flat_border=0x00808080
bkc_control_def=0x00C0C0C0
fgc_control_normal=0x00000000
fgc_control_disabled=0x00C0C0C0
bkc_hilight_normal=0x00FF0000
bkc_hilight_lostfocus=0x00BDA69C
fgc_hilight_normal=0x00FFFFFF
fgc_hilight_disabled=0x00C0C0C0
bkc_desktop=0x00FF0000
bkc_dialog=0x00C0C0C0
bkc_tip=0x00C8FCF8

[imeinfo]
imetabpath=/usr/local/lib/minigui/res/imetab/
imenumber=1
ime0=pinyin

[appinfo]
apprespath=/usr/local/lib/shared/miniguiapps/
```

在使用默认配置安装 MiniGUI 之后，将把 MiniGUI 源代码树中的 etc/MiniGUI-3d.cfg 文件安装到系统 /usr/local/etc/ 目录，并更名为 MiniGUI.cfg。在 MiniGUI 应用程序启动时，MiniGUI 优先查找用户主目录下的 .MiniGUI.cfg 文件，其次是 /usr/local/etc/MiniGUI.cfg，最后是 /etc/MiniGUI.cfg 文件。如果用户没有在自己的主目录下建立 .MiniGUI.cfg 文件，则通常情况下，/usr/local/etc/MiniGUI.cfg 文件就是 MiniGUI 所使用的默认运行时配置文件。

该配置文件采用了非常简洁的格式，所以修改起来也很容易。其格式如下：

```
[section-name1]
key-name1=key-value1
key-name2=key-value2

[section-name2]
key-name3=key-value3
key-name4=key-value4
```

该配置文件中的参数以段（section）分组，然后用 key=value 的形式指定参数及其值。

下面介绍一些重要的段。

4.2.1 system

system 段中指定了 MiniGUI 要使用的图形引擎、输入引擎以及鼠标设备和协议类型，分别由 gal_engine、ial_engine、mdev 和 mtype 键指定。因为 MiniGUI 库中可以同时包含多个图形引擎和多个输入引擎，可以分别通过 gal_engine、ial_engine 指定要使用哪个图形引擎。mdev 和 mtype 分别用来指定鼠标设备文件以及鼠标协议类型。

4.2.2 fbcon 和 qvfb

fbcon 段的 defaultmode 关键字定义使用 FBCON 图形引擎时默认的显示模式。

qvfb 段的 defaultmode 关键字定义使用基于 NEWGAL 接口的 qvfb 引擎时默认的显示模式；display 关键字指定运行 qvfb 时使用了 X Window 的哪个 Display，一般取 0。

4.2.3 systemfont

systemfont 段定义了 MiniGUI 的系统字体，默认情况下不应作改动。font_number 指定了要创建的系统字体个数，然后用 name<NR> 表示编号为 <NR> 的系统字体名称。字体名的格式如下：

```
<type>-<facename>-<style>-<width>-<height>-<charset1>
```

需要说明的是，系统字体是 MiniGUI 装载了由 rawbitmapfonts、varbitmapfonts、qpf、truetypefonts 、 t1fonts 等段定义的设备字体之后，根据上述字体名称调用 CreateLogFontFromName 函数建立的逻辑字体。这些字体将用于 MiniGUI 的标题、菜单、控件的显示，同时还用来指定窗口的默认字体。

逻辑字体名称各部分的含义如下：

- type 是所选用的设备字体类型，如果不想指定，则取 *。
- facename 指字体样式名，比如 Courier、Times 等等。
- style 是由六个字母组成的字符串，用来指定逻辑字体风格，比如是否斜体、是否加粗、是否含有下划线或者删除线等等。
- width 指定要创建的逻辑字体的宽度。一般不用指定，取 *。
- height 指定要创建的逻辑字体的高度。
- charset 指定要创建的逻辑字体的字符集。

此外，在 systemfont 段中，还有下面的键需要定义，这些键的值就是上述逻辑字体的编号：

```
default=0
wchar_def=1
fixed=1
caption=2
menu=3
control=3
```

举例来说，上面的键定义了系统（单字节字符集的）默认字体为编号为 0 的系统字体；多字节字符集的默认字体是 1 号系统字体；等宽字体为 1 号系统字体；窗口标题使用 2 号系统字体；菜单使用 3 号系统字体；控件使用 3 号系统字体。

您可以修改要创建的系统字体个数，但至少要创建一种单字节字符集（比如 ISO8859-1）的字体。还可以修改系统字体的大小，比如，下面的 systemfont 就指定 MiniGUI 使用 16 点阵的系统字体，并只创建了用于显示 ISO8859-1 字符集和 GB2312-80 字符集的逻辑字体，然后用这两个逻辑字体显示标题、菜单和控件中的文字：

```
[systemfont]
font_number=2
name0=rbf-Fixed-rrncnn-8-16-ISO8859-1
name1=rbf-Fixed-rrncnn-16-16-GB2312.1980-0

default=0
wchar_def=1
fixed=1
caption=1
menu=1
control=1
```

需要进一步说明的是：

- MiniGUI 根据 default、wchar_def 系统字体来定义系统的默认字符集，这影响 GetSysCharset、GetSysCharWidth、GetSysCCharWidth 和 GetSysHeight 函数的返回值。一般来讲，default 和 wchar_default 必须是等宽点阵字体，即 RBF 字体，并且多字节字符集字体的宽度必须是单字节字符集宽带的两倍。
- MiniGUI 的许多窗口元素尺寸是根据系统字体的大小定义的，详情请参阅下面对 mainwinmetric 段的说明。

随增值版产品发布的字体比较丰富，因此，增值版用户的 [systemfont] 段定义如下：

```
# The system fonts must be raw bitmap fonts
[systemfont]
font_number=6
font0=rbf-fixed-rrncnn-6-12-ISO8859-1
font1=rbf-fixed-rrncnn-12-12-GB2312
font2=*-Courier-rrncnn-*-12-GB2312
font3=*-SansSerif-rrncnn-*-12-GB2312
font4=*-Times-rrncnn-*-12-GB2312
font5=*-Helvetica-rrncnn-*-12-GB2312

default=0
```

```
wchar_def=1
fixed=1
caption=2
menu=3
control=3
```

4.2.4 rawbitmapfonts、varbitmapfonts、qpf、truetypefonts 和 type1fonts

这些段用来指定要装载的设备字体信息。段中的 font_number 键定义了要装载的设备字体个数；name<NR> 和 fontfile<NR> 键分别定义了编号为 <NR> 的设备字体名称及对应的字体文件。如果您不想装载某个类型的设备字体，则可以通过设置 font_number 为 0 值来跳过对这种设备字体的装载。MiniGUI 使用的设备字体名称格式如下：

```
<type>-<facename>-<style>-<width>-<height>-<charset1[,charset2,...]>
```

设备字体名称各部分的含义如下：

- type 是设备字体类型，对上述四种设备字体，分别取 rbf、vbf、qpf、ttf 和 t1f。
- facename 指设备字体的样式名，比如 Courier、Times 等等。
- style 是由六个字母组成的字符串，用来指定字体风格，比如是否斜体、是否加粗、是否含有下划线或者删除线等等。
- width 表示字体的宽度。对变宽字体来讲，用来指定最大宽度。对可缩放的矢量字体来讲，设置为 0。
- height 表示字体的高度。对可缩放的矢量字体来讲，设置为 0。
- charset1、charset2 等，表示该设备字体所支持的字符集。

增值版产品中，上述三个段的定义如下：

```
[rawbitmapfonts]
font_number=4
name0=rbf-fixed-rrncnn-8-16-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/8x16-iso8859-1.bin
name1=rbf-fixed-rrncnn-8-16-GB2312.1980-0
fontfile1=/usr/local/lib/minigui/res/font/song-16-gb2312.bin
name2=rbf-fixed-rrncnn-6-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/6x12-iso8859-1.bin
name3=rbf-fixed-rrncnn-12-12-GB2312.1980-0
fontfile3=/usr/local/lib/minigui/res/font/song-12-gb2312.bin

[varbitmapfonts]
font_number=7
name0=vbf-Courier-rrncnn-8-13-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/Courier-rr-8-13.vbf
name1=vbf-Helvetica-rrncnn-11-12-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/Helvetica-rr-11-12.vbf
name2=vbf-Times-rrncnn-10-12-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/Times-rr-10-12.vbf
name4=vbf-Courier-rrncnn-10-15-ISO8859-1
fontfile4=/usr/local/lib/minigui/res/font/Courier-rr-10-15.vbf
name5=vbf-Helvetica-rrncnn-15-16-ISO8859-1
```

```
fontfile5=/usr/local/lib/minigui/res/font/Helvetica-rr-15-16.vbf
name6=vbf-Times-rrncnn-13-15-ISO8859-1
fontfile6=/usr/local/lib/minigui/res/font/Times-rr-13-15.vbf

[qpf]
font_number=4
name0=qpf-fixed-rrncnn-16-16-ISO8859-1,ISO8859-15,GB2312,GBK,BIG5
fontfile0=/usr/local/lib/minigui/res/font/unifont_160_50.qpf
name1=qpf-times-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile1=/usr/local/lib/minigui/res/font/smoothtimes_100_50.qpf
name2=qpf-helvetica-rrncnn-5-10-ISO8859-1,ISO8859-15
fontfile2=/usr/local/lib/minigui/res/font/helvetica_100_50.qpf
name3=qpf-micro-rrncnn-4-4-ISO8859-1,ISO8859-15
fontfile3=/usr/local/lib/minigui/res/font/micro_40_50.qpf

[truetypefonts]
font_number=3
name0=ttf-arial-rrncnn-0-0-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/arial.ttf
name1=ttf-times-rrncnn-0-0-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/times.ttf
name2=ttf-pinball-rrncnn-0-0-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/pinball.ttf

[type1fonts]
font_number=9
name0=type1-Charter-rrncnn-0-0-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/bchr.pfb
name1=type1-Charter-rincnn-0-0-ISO8859-1
fontfile1=/usr/local/lib/minigui/res/font/bchri.pfb
name2=type1-Charter-brncnn-0-0-ISO8859-1
fontfile2=/usr/local/lib/minigui/res/font/bchb.pfb
name3=type1-Charter-bincnn-0-0-ISO8859-1
fontfile3=/usr/local/lib/minigui/res/font/bchbi.pfb
name4=type1-Courier-rrncnn-0-0-ISO8859-1
fontfile4=/usr/local/lib/minigui/res/font/dcr10.pfb
name5=type1-Courier-rincnn-0-0-ISO8859-1
fontfile5=/usr/local/lib/minigui/res/font/dcti10.pfb
name6=type1-Courier-brncnn-0-0-ISO8859-1
fontfile6=/usr/local/lib/minigui/res/font/dcbx10.pfb
name7=type1-Courier-bincnn-0-0-ISO8859-1
fontfile7=/usr/local/lib/minigui/res/font/dcbxti10.pfb
name8=type1-eufm10-rrncnn-0-0-ISO8859-1
fontfile8=/usr/local/lib/minigui/res/font/eufm10.pfb
```

4.2.5 mouse、event

这两个段用于系统的内部事件处理，一般无须作任何改动。

4.2.6 cursorinfo、iconinfo 和 bitmapinfo

指定 MiniGUI 要装载的鼠标光标、图标以及系统位图信息。这三个段的定义方法和 rawbitmapfonts 等段类似。首先分别用 cursorpath、iconpath 和 bitmappath 指定了存放鼠标光标、图标和位图的路径，然后用 cursornumber、iconnumber 和 bitmapnumber 分

别指定了要装载的鼠标光标、图标和位图的个数，最后用 cursor<NR>、icon<NR>、bitmap<NR> 指定了编号为 <NR> 的鼠标光标、图标和位图文件。

需要注意的是，您可以修改 cursornumber、iconnumber、bitmapnumber 来减少 MiniGUI 装载的鼠标光标、图标以及位图文件，并同时删除对应的哪些资源文件，这样，可以减少 MiniGUI 的存储空间占用量。当然，在减少这些资源之前，您要确保自己的应用程序真的不需要这些资源。

如果您在配置 MiniGUI 时使用了 --disable-cursor 选项，则 MiniGUI 会忽略 cursorinfo 段。

bitmap6 用来指定 MiniGUI 的背景图片。如果您不需要背景图片，则可以修改 bitmap6 的值，将其指定为 none。bitmapinfo 中还有若干特殊的键，比如 IMEctrlbtn 和 logo；前者由输入法模块使用，后者由 About 对话框使用。如果您在配置 MiniGUI 时使用了 --disable-imegb2312 或者 --disable-aboutdlg 选项，则可以删除这两个键定义的位图。

4.2.7 bgpicture

这个段定义了背景图片在背景上显示时的位置，可取 center、upleft、downleft、upright、downright、upcenter、downcenter、vcenterleft、vcenterright、none 这几个值，分别表示正中、左上、左下、右上、右下、中上、中下、左中、右中、不显示等。

4.2.8 mainwinmetrics 和 windowelementcolors

mainwinmetrics 定义了默认的窗口元素的尺寸，一般不需要进行修改。我们可以使用系统字体的高度来设定窗口元素的尺寸。比如在指定标题栏的高度时，用 captiony=+4 这样的形式指定标题栏高度是系统字体高度加 4 个象素。windowelementcolors 定义了默认的窗口元素所使用的颜色，一般也不需要进行修改。

4.2.9 imeinfo

这个段指定了 GB2312 输入法要装载的输入法个数及对应的模块。键的名称规则类似 bitmapinfo 段。pinyin、wubi、shuangpin、ziranma 等分别表示全拼、五笔、双拼、自然码等输入法模块。如果在配置 MiniGU 时打开了 GB2312 输入法（--enable-imegb2312），则会根据 imenumber 的值装载指定的输入法模块；如果 imenumber 的值为 0，则输入法模块仅仅提供内码输入法。

4.2.10 支持中文显示的最小配置文件

MiniGUI 中包含一个支持中文显示的最小配置文件（etc/MiniGUI-min.cfg），使用该配置文件，将使运行时的 MiniGUI 占用最少的内存。该配置文件主要修改了字体和鼠标光标配置段，可配合 4.1.3 中提到的配置脚本一同使用。该文件的内容如下：

```
# MiniGUI Ver 1.6.x
# This configuration file is for minimal configuration.
#
# Copyright (C) 2002~2004 Feynman Software.
# Copyright (C) 1998~2002 Wei Yongming.
#
# Web:   http://www.minigui.com
# Web:   http://www.minigui.org
#
# This configuration file must be installed in /etc,
# /usr/local/etc or your home directory. When you install it in your
# home directory, the name should be ".MiniGUI.cfg".
#
# The priority of above configruation files is ~/.MiniGUI.cfg,
# /usr/local/etc/MiniGUI.cfg, and then /etc/MiniGUI.cfg.
#
# If you change the install path of MiniGUI resource, you should
# modify this file to meet your configuration.
#
# NOTE:
# The format of this configuration file has changed since last release.
# Please DONT forget to provide the latest MiniGUI.cfg file for your MiniGUI.
#

[system]
# GAL engine
gal_engine=fbcon

# IAL engine
ial_engine=console

mdev=/dev/mouse
mtype=IMPS2

[fbcon]
defaultmode=1024x768-16bpp

[qvfb]
defaultmode=640x480-16bpp
display=0

# The system fonts must be raw bitmap fonts
[systemfont]
font_number=1
font0=rbf-fixed-rrncnn-8-16-ISO8859-1

default=0
wchar_def=0
fixed=0
caption=0
```

```
menu=0
control=0

[rawbitmapfonts]
font_number=1
name0=rbf-fixed-rrncnn-8-16-ISO8859-1
fontfile0=/usr/local/lib/minigui/res/font/8x16-iso8859-1.bin

[varbitmapfonts]
font_number=0

[qpf]
font_number=0

[truetypefonts]
font_number=0

[type1fonts]
font_number=0

[mouse]
dblclicktime=300

[event]
timeoutusec=300000
repeatusec=50000

[cursorinfo]
# Edit following line to specify cursor files path
cursorpath=/usr/local/lib/minigui/res/cursor/
cursornumber=2
cursor0=d_arrow.cur
cursor1=d_beam.cur

[iconinfo]
# Edit following line to specify icon files path
iconpath=/usr/local/lib/minigui/res/icon/
# Note that max number defined in source code is 5.
iconnumber=5
icon0=form-flat.ico
icon1=w95mbx01-flat.ico
icon2=w95mbx02-flat.ico
icon3=w95mbx03-flat.ico
icon4=w95mbx04-flat.ico

# default icons for TREEVIEW control
fold=fold.ico
unfold=unfold.ico

# default icons for new OpenFileDialogBox
dir=folder.ico
file=textfile.ico

[bitmapinfo]
# Edit following line to specify bitmap files path
bitmappath=/usr/local/lib/minigui/res/bmp/
# Note that max number defined in source code is 7
bitmapnumber=6
```

```
bitmap0=capbtns.bmp
bitmap1=arrows.bmp
bitmap2=none
bitmap3=none
bitmap4=none
bitmap5=none
# background picture, use your favirate photo
bitmap6=none

# bitmap used by BUTTON control
button=button.bmp
pushbutton=none
pushedbutton=none

# bitmap used by LISTBOX control
checkmark=checkmark.bmp

# bitmap used by COMBOBOX control
downarrow=downarrow.bmp
updownarrow=updownarrow.bmp
leftrightarrow=leftrightarrow.bmp

# bitmap used by SPINBOX control
spinbox=spinbox.bmp

# bitmap used by listview control
lvfold=lvfold.bmp
lvunfold=lvunfold.bmp

# bitmaps used by IME window
IMEctrlbtn=shurufa-flat.bmp

logo=MiniGUI256.bmp
# logo=MiniGUI16.bmp

[bgpicture]
position=center
# position=upleft
# position=downleft
# position=upright
# position=downright
# position=upcenter
# position=downcenter
# position=vcenterleft
# position=vcenterright
# position=none
```

如果要使用该配置文件，可将其复制到 /usr/local/etc/ 目录下，并覆盖该目录下的 MiniGUI.cfg 文件；或者，您也可以将这个文件复制到自己的主目录下 ~/.MiniGUI.cfg，这样，就会优先使用该配置文件。

需要注意的是，上面给出的这个配置文件省略了部分未作修改的段，比如 [mainwinmetrics] 以及 [windowelementcolors] 等。

4.2.11 使用内建式资源时 MiniGUI 的配置

在使用内建式资源（incore resources）时，MiniGUI 不需要 MiniGUI.cfg 配置文件，相应的配置选项在 src/sysres/mgetc.c 文件中指定。下面是该文件的内容：

```
/*
 ** $Id: mgetc.c,v 1.73 2004/07/28 01:31:30 weiym Exp $
 **
 ** mgetc.c: definitions for incore resource.
 **
 ** Copyright (C) 2003 Feynman Software.
 **
 ** Create date: 2003/09/22
 */

/*
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*/

#include "common.h"

typedef struct _ETCSECTION
{
    int key_nr;               /* key number in the section */
    char *name;               /* name of the section */
    char **keys;              /* key string arrays */
    char **values;            /* value string arrays */
} ETCSECTION;
typedef ETCSECTION* PETCSECTION;

typedef struct _ETC_S
{
    int section_nr;           /* number of sections */
    PETCSECTION sections;     /* pointer to section arrays */
} ETC_S;


#ifdef _INCORE_RES

static char *SYSTEM_KEYS[] = {"gal_engine", "ial_engine", "mdev", "mtype"};

#ifdef __ECOS__
static char *SYSTEM_VALUES[] = {"ecoslcd", "palm2", "/dev/ts", "none"};
#elif defined (__UCOSII__)
static char *SYSTEM_VALUES[] = {"ucos2lcd", "ucos2", "/dev/mouse", "IMPS2"};
```

```
#elif defined (_NEWGAL_ENGINE_BF533)
static char *SYSTEM_VALUES[] = {"bf533", "hh5249kbdir", "/dev/mouse", "IMPS2"};
static char *BF533_KEYS[] = {"defaultmode"};
static char *BF533_VALUES[] = {"640x480-16bpp"};
#elif defined (_NEWGAL_ENGINE_UTPMC)
static char *SYSTEM_VALUES[] = {"utpmc", "utpmc", "", ""};
static char *UTPMC_KEYS[] = {"defaultmode"};
static char *UTPMC_VALUES[] = {"320x240-16bpp"};
#elif defined (__VXWORKS__)
static char *SYSTEM_VALUES[] = {"commlcd", "comm", "/dev/ts", "none"};
#elif defined (_L7200_IAL)
static char *SYSTEM_VALUES[] = {"fbcon", "l7200", "/dev/tscreen", "none"};
#else
//static char *SYSTEM_VALUES[] = {"fbcon", "console", "/dev/mouse", "IMPS2"};
static char *SYSTEM_VALUES[] = {"qvfb", "qvfb", "/dev/mouse", "IMPS2"};
#endif

#ifdef __VXWORKS__
static char *COMMLCD_KEYS[] = {"defaultmode"};
static char *COMMLCD_VALUES[] = {"106x120-16bpp"};
#endif

static char *FBCON_KEYS[] = {"defaultmode"};
static char *FBCON_VALUES[] = {"640x480-16bpp"};

static char *DUMMY_KEYS[] = {"defaultmode"};
static char *DUMMY_VALUES[] = {"640x480-16bpp"};

static char *QVFB_KEYS[] = {"defaultmode", "display"};
static char *QVFB_VALUES[] = {"640x480-16bpp", "0"};

#ifdef __ECOS__

static char *SYSTEMFONT_KEYS[] =
{"font_number", "font0", "font1", "font2", "default", "wchar_def", "fixed", "caption"
, "menu", "control"};

static char *SYSTEMFONT_VALUES[] =
{
    "3","rbf-fixed-rrncnn-12-24-ISO8859-1", "vbf-Helvetica-rrncnn-*-27-ISO8859-1", "v
bf-Helvetica-rrncnn-*-21-ISO8859-1",
    "0", "0", "0", "2", "2", "1"
};

#elif defined (_PHONE_WINDOW_STYLE)

static char *SYSTEMFONT_KEYS[] =
{"font_number", "font0", "font1", "font2", "font3", "default", "wchar_def", "fixed",
"caption", "menu", "control"};

static char *SYSTEMFONT_VALUES[] =
{
    "4", "vbf-Helvetica-rrncnn-*-21-ISO8859-1", "vbf-Helvetica-rrncnn-*-21-GB2312", "
*-Helvetica-rrncnn-*-16-GB2312", "*-fixed-rrncnn-*-16-GB2312",
    "0", "1", "3", "2", "2", "2"
};

#elif defined (__VXWORKS__)
```

```
static char *SYSTEMFONT_KEYS[] =
{"font_number", "font0", "font1", "font2", "font3", "default", "wchar_def", "fixed",
"caption", "menu", "control"};

static char *SYSTEMFONT_VALUES[] =
{
    "4","rbf-fixed-rrncnn-6-12-ISO8859-1", "rbf-gothic-rrncnn-6-12-JISX0201-1", "rbf-
gothic-rrncnn-12-12-JISX0208-1",
        "*-fixed-rrncnn-*-16-JISX0208-1",
    "0", "2", "2", "2", "3", "2"
};

#else

static char *SYSTEMFONT_KEYS[] =
{"font_number", "font0", "font1", "font2", "default", "wchar_def", "fixed", "caption"
, "menu", "control"};

static char *SYSTEMFONT_VALUES[] =
{
    "3","rbf-fixed-rrncnn-6-12-ISO8859-1", "*-fixed-rrncnn-*-16-GB2312", "*-Helvetica
-rrncnn-*-16-GB2312",
    "0", "1", "1", "2", "2", "2"
};
#endif

static char *CURSORINFO_KEYS[] = {"cursornumber"};
static char *CURSORINFO_VALUES[] = {"3"};

static char *ICONINFO_KEYS[] = {"iconnumber"};
static char *ICONINFO_VALUES[] = {"5"};

static char *BITMAPINFO_KEYS[] = {"bitmapnumber"};
static char *BITMAPINFO_VALUES[] = {"7"};

#ifdef _SVPXX_IAL
static char *EVENT_KEYS[] = {"timeoutusec", "repeatusec"};
static char *EVENT_VALUES[] = {"100000", "50000"};
#endif

static char *BGPICTURE_KEYS[] = {"position"};
static char *BGPICTURE_VALUES[] = {"none"};
/*

static char *MOUSE_KEYS[] = {"dblclicktime"};
static char *MOUSE_VALUES[] = {"300"};

static char *EVENT_KEYS[] = {"timeoutusec", "repeatusec"};
static char *EVENT_VALUES[] = {"300000", "50000"};

*/

static ETCSECTION mgetc_sections [] =
{
  {4, "system",        SYSTEM_KEYS,      SYSTEM_VALUES},
  {1, "fbcon",         FBCON_KEYS,       FBCON_VALUES},
  {1, "em85xxosd",     FBCON_KEYS,       FBCON_VALUES},
```

```
  {1, "dummy",          DUMMY_KEYS,       DUMMY_VALUES},
#ifdef __ECOS__
  {1, "ecoslcd",        FBCON_KEYS,       FBCON_VALUES},
#elif defined(__UCOSII__)
  {1, "ucos2lcd",       FBCON_KEYS,       FBCON_VALUES},
#elif defined(__WINBOND_SWLINUX__)
  {1, "svpxxosd",       FBCON_KEYS,       FBCON_VALUES},
#elif defined (_NEWGAL_ENGINE_BF533)
  {1, "bf533",          BF533_KEYS,       BF533_VALUES},
#elif defined (_NEWGAL_ENGINE_UTPMC)
  {1, "utpmc",          UTPMC_KEYS,       UTPMC_VALUES},
#elif defined(__VXWORKS__)
  {1, "commlcd",        COMMLCD_KEYS,     COMMLCD_VALUES},
#endif
  {2, "qvfb",           QVFB_KEYS,        QVFB_VALUES},
  {TABLESIZE(SYSTEMFONT_KEYS),"systemfont",    SYSTEMFONT_KEYS, SYSTEMFONT_VALUES},
  {1, "cursorinfo",     CURSORINFO_KEYS, CURSORINFO_VALUES},
  {1, "iconinfo",       ICONINFO_KEYS,    ICONINFO_VALUES},
  {1, "bitmapinfo",     BITMAPINFO_KEYS, BITMAPINFO_VALUES},
/* optional sections */
  {1, "bgpicture", BGPICTURE_KEYS, BGPICTURE_VALUES},
  /*
  {1, "mouse", MOUSE_KEYS, MOUSE_VALUES},
  */
#ifdef _SVPXX_IAL
  {2, "event", EVENT_KEYS, EVENT_VALUES},
#endif
};

ETC_S MGETC = { TABLESIZE (mgetc_sections), mgetc_sections };

#endif /* _INCORE_RES */
```

和 MiniGUI.cfg 配置文件的结构类似，mgetc.c 中定义了一个 ETC_S 类型的结构 MGETC 和 ETCSECTION 类型的结构数组 mgetc_sections，指明了 MiniGUI 中的各项系统配置信息。

MGETC 结构的第一项指定段（section）的个数，第二项为 ETCSECTION 类型的结构数组，其元素个数应不小于第一项指定的值。这里指定段的个数为 TABLESIZE(mgetc_sections)，段数组为 mgetc_sections。

ETCSECTION 段结构的第一项指定段中键（key）的个数，第二项指定该段的名称，第三段和第四段分别指定键（key）数组和值（value）数组。键数组和值数组中元素的个数应和 ETCSECTION 结构的第一项指定的键个数一致。

mgetc_sections 数组中定义的段基本上都是必须的（fbcon 和 qvfb 可只定义适用的一个），其它注释掉的段是可选的。这几个段的含义和 MiniGUI.cfg 配置文件中相应段的含义是一样的。用户一般情况下只需修改“system”段和“fbcon”段，指定 MiniGUI 的 GAL 引擎和显示模式，以及 IAL 引擎。

“systemfont”段定义了系统使用的内建式字体，MiniGUI 1.6.x 目前支持 ISO8859-1 和 GB2312 字符集的 12 点阵 RBF 字体。在采用内建式资源时，QPF 字体、TTF 字体及 Type1 字体不被支持。

5 MiniGUI 的交叉编译

为了能让 MiniGUI 运行在不同的目标平台上，需要有针对相应平台的交叉编译工具。交叉编译工具是运行在开发平台（一般是 x86/Linux）上、用于生成目标平台上的可执行代码的工具集，一般包含：binutils 二进制工具、gcc 交叉编译器、glibc 库。binutils 包含如 as 汇编器、ld 链接器等操纵目标文件的二进制工具，Glibc 是 GNU 的标准 C 函数库，gcc 则是编译程序时要用到的 gcc 交叉编译器。

一般而言，安装完 binutils、gcc 和 glic 等交叉编译工具之后，需要设定一定的环境参数（如 PATH）之后，才可在宿主开发平台上编译程序。编译程序时指定编译器为所要使用的交叉编译器，此时 make 工具编译程序所使用的编译工具和标准函数库都是交叉编译工具链所提供的，而不是开发平台系统本身自带的 x86 编译工具，编译生成的目标代码是目标平台格式的代码和文件。

利用 GNU Autoconf/Automake 系统所提供的对交叉编译的有力支持，按照类似如下的步骤进行相关的设置，就能非常方便地完成 MiniGUI 和应用程序的交叉编译工作。编译完成之后，把所生成的目标平台格式的 MiniGUI 库和相关的头文件安装到交叉编译环境中，就可以进行 MiniGUI 程序的交叉编译了。

下面以 StrongArm 平台为例讲述 MiniGUI 和应用程序的交叉编译。

5.1 安装和设置交叉编译环境

我们以 arm-toolchain-2.1.3 工具链为交叉编译工具，该工具链可用于 StrongARM 平台的交叉编译，必须安装的基本工具包括：

- arm-linux-binutils-2.10-1.i386.rpm
- arm-linux-gcc-2.95.2-1.i386.rpm
- arm-linux-glibc-2.1.3-1.i386.rpm

这些包可以从 INTERNET 上下载, 执行 rpm –ivh *.rpm 命令，就可以将这些包安装到系统上。在此例中，这些工具将被安装在路径 /usr/local/arm-linux 下。arm-linux-binutils-2.10-1.i386.rpm 中包括 arm-strip、arm-ar 等二进制工具，这些命令的功能和没有 arm- 前缀的 PC 命令一样；arm-linux-glibc-2.1.3-1.i386.rpm 中包含的是标准 C 函数库 libc6 的 ARM 版本，以及对应的 C 头文件；arm-linux-glibc-2.1.3-1.i386.rpm 中包含的是生成 ARM 平台代码的 gcc 交叉编译器。

为了让 Shell 和 make 工具能正确地找到安装在 /usr/local/arm-linux 下的编译器及二进制工具，需要修改 PATH 环境变量，将 gcc 交叉编译器所在的目录（这里就是 /usr/local/arm-linux/bin）添加到 PATH 中去。比如，您可以在自己主目录的 .bash_profile 文件中如下设置 PATH 环境变量：

```
export PATH=/usr/local/arm-linux/bin:$PATH
```

5.2 配置 MiniGUI

首先请确保您的系统中安装了 automake 和 autoconf 工具。进入 MiniGUI 的源代码树，运行 autogen.sh 脚本，重新生成 configure 脚本，并删除 config.cache、config.status 文件：

```
$ ./autogen.sh
$ rm config.cache config.status
```

如果已经生成了 configure 脚本，就不需要上述步骤了。

运行 ./configure 脚本，注意指定正确的目标平台类型和函数库的安装路径：

```
$ CC=arm-linux-gcc \
  ./configure --target=arm-linux --prefix=/usr/local/arm-linux/arm-linux
```

arm-linux-gcc 是 StrongArm 的交叉编译器，在第一步安装 RPM 包时被安装到 /usr/local/arm-linux/bin 下。--prefix 选项的含义已经在本书的第 3 章中介绍了，这里需要注意的是，prefix 应该是所使用的交叉编译环境中系统头文件目录 include 和库目录 lib 所在的目录，在本例中就是/usr/local/arm-linux/arm-linux。这里 MiniGUI 的函数库和头文件也将被分别安装到交叉编译环境中的 /usr/local/arm-linux/arm-linux 目录下的 lib 和 include 目录中。这样，在进行后续的 MiniGUI 应用程序的交叉编译时，就可以正确地找到 MiniGUI 的头文件和函数库，否则，编译 MiniGUI 应用程序时就需要通过-I 和-L 参数指定 MiniGUI 头文件和库搜索路径。

MiniGUI 增值版中的 menuconfig 配置工具已经内置了各种交叉编译工具链的支持，用户使用 make menuconfig 命令进入图形配置界面，然后选择相应的交叉编译器，就可以对 MiniGUI 进行相应的目标平台的交叉编译。具体操作见第 6 章 “使用 MiniGUI 增值版中的嵌入式 Linux 开发工具”。

5.3 编译和安装

配置完成之后，输入命令 make 就可以对 MiniGUI 进行交叉编译。我们已经知道，只要指定编译器 CC=arm-linux-gcc，编译 MiniGUI 所使用的编译工具、头文件、函数库，都将来自 /usr/local/arm-linux 目录，而不是主机的 /usr 目录。

make 编译完成之后，使用 make install 把 MiniGUI 的函数库和头文件以及配置文件等资源被安装到交叉编译环境中，在本例中就是/usr/local/arm-linux/arm-linux 目录:

- 函数库：/usr/local/arm-linux/arm-linux/lib/
- 头文件：/usr/local/arm-linux/arm-linux/include/
- 手册页：/usr/local/arm-linux/arm-linux/man/
- 配置文件：/usr/local/arm-linux/arm-linux/etc/
- MiniGUI 的资源：/usr/local/arm-linux/arm-linux/lib/minigui

5.4 目标系统和运行

一般而言，在嵌入式系统开发过程中，我们编译完 MiniGUI 和应用程序之后，需要把 MiniGUI 库、资源和应用程序拷贝到为目标系统准备的文件系统（以下简称目标文件系统）目录中，然后使用相关的工具生成目标映像，再下载到目标板上运行。在某些目标系统上，您也可以使用某些下载工具通过串口或以太网口单独下载文件到目标系统中。

如果您配置 MiniGUI 时选择嵌入资源（--enable-incoreres），MiniGUI 运行需要一些位图、图标和字体等资源将被编译到代码中，运行时就不再需要单独的资源文件了。

如果不内嵌资源（--enable-incoreres 选项）的话，您需要把 MiniGUI 运行所需的资源（来自 minigui-res-1.6.x 的资源和其它资源）拷贝到为目标文件系统中，资源存放的位置根据目标系统的特性来确定，一般来说可以存放在系统中只读的数据存储区。可以修改 MiniGUI.cfg 配置文件来指定资源的搜索路径，当然，MiniGUI.cfg 配置文件也需要放到目标文件系统的某个目录下。

通常我们先在 x86 PC 上的 MiniGUI 开发环境中进行程序开发、运行和调试，然后在下载到目标系统中运行和调试，所以，只需把 x86 平台上 MiniGUI 运行所需的资源（一般是/usr/local/lib/minigui 目录）和/usr/local/etc/目录下的配置文件拷贝到目标文件系统中的相应目录，删除掉不需要的资源，然后适当地修改 MiniGUI.cfg 配置文件就可以了。

可以将 minigui-res 包中的资源直接安装到目标文件系统的某个目录下。进入 minigui-res-1.6.x 目录并编辑 config.linux 文件，将其中的 prefix 变量修改为：

```
prefixe=<TARGET_DIR>
```

然后执行 make install 即可。

5.5 关于 MDE 的交叉编译和运行

我们也可以参照上面的步骤对 MiniGUI 的演示程序 MDE 进行交叉编译，因为过程类

似，不再赘述。不过在真正将 MDE 程序复制到目标平台上运行时，您需要注意如下问题：

- MDE 的 mginit 程序默认启动 vcongui 程序。这个程序需要伪终端功能的支持，因此，需要在目标系统上建立 /dev/ttyp5、/dev/ptyp5 这样的伪终端设备。或者，可以修改 mginit.rc 程序，将默认启动的程序修改为其它程序，比如 bomb/ 目录下的 bomb 扫雷程序，这时，需要修改 mginit.rc 配置文件 [mginit] 段中的 autostart 值，使之等于 1。
- MDE 中的 mginit 在启动时需要装载表示应用程序的图标，有些应用程序图标是 PNG 格式的。如果您的目标系统上未安装 libpng 库，则 mginit 会在初始化时出现错误而退出。这时，您可以减小 mginit.rc 配置文件中 [mginit] 段中的 nr 值，使之小于 8。

因为以上的一些限制，我们建议用户在嵌入式开发板上运行 MiniGUI 程序时使用 mg-samples 软件包。该软件包中的示例程序相对简单，对系统的要求不高。有关 mg-samples 中示例程序的详细说明，请参阅《MiniGUI 编程指南》。

6 使用增值版中的嵌入式开发工具

MiniGUI 1.3 增值版中的嵌入式开发软件包括如下内容：

- MiniGUI 1.6.x 源代码及演示程序包 MDE、示例程序包 samples 以及其他资源包。
- 图形界面配置工具
- 交叉编译工具链
- MiniGUI 相关依赖库的源代码
- uClinux-dist 软件包和其它工具

利用 MiniGUI 增值版包含的这些嵌入式 Linux 开发工具和软件包，您可以快速、方便地搭建好相关平台的嵌入开发环境。

6.1 交叉编译工具的安装和使用

MiniGUI 1.3 增值版提供了针对 ARM、MIPS、PowerPC 、m68k、coldfire 等平台的交叉编译工具：

- arm-toolchain-2.1.3 交叉编译工具链
- armv4l 交叉编译工具链
- mipsel 交叉编译工具链
- ppc 交叉编译工具链
- m68k-elf 交叉编译工具链，用于 uClinux
- arm-elf 交叉编译工具链，用于 uClinux

上述交叉编译工具分别存放在增值版产品 CD-ROM 的如下目录中：

- cross-tools/arm/arm-2.1.3
- cross-tools/arm/armv4l
- cross-tools/mips
- cross-tools/ppc
- cross-tools/m68k-elf
- cross-tools/arm-elf

6.1.1 arm 交叉编译工具链

arm-toolchain-2.1.3 交叉编译工具链适用于 StrongARM、xScale 等处理器，包括如下的编译工具：

- arm-linux-binutils-2.10-1.i386.rpm
- arm-linux-gcc-2.95.2-1.i386.rpm

- arm-linux-glibc-2.1.3-1.i386.rpm

安装和使用方法如下：

1. 安装交叉编译器

```
# rpm -ivh arm-linux-binutils-2.10-1.i386.rpm
# rpm -ivh arm-linux-gcc-2.95.2-1.i386.rpm
# rpm -ivh arm-linux-glibc-2.1.3-1.i386.rpm
```

以上命令将把 arm-toolchain-2.1.3 工具链安装到/usr/local/arm-linux 目录。

2. 交叉编译

首先以 root 登录，然后设置 PATH 环境变量：

```
# export PATH=/usr/local/arm-linux/bin:$PATH
```

然后编译程序时把 CC 设为 arm-linux-gcc 就可以进行编译了。

6.1.2 armv4l 交叉编译工具链

armv4l-tools 交叉编译工具链提供如下的编译工具和库：

- cross-armv4l-binutils-2.10-3mz.i386.rpm
- cross-armv4l-gcc-2.95.2-10mz.i386.rpm
- cross-armv4l-gcc-c++-2.95.2-10mz.i386.rpm
- cross-armv4l-gdb-5.2.1-1mz.i686.rpm
- cross-armv4l-glib-1.2.10-1mz.i386.rpm
- cross-armv4l-glib-devel-1.2.10-1mz.i386.rpm
- cross-armv4l-glibc-2.2.1-2mz.i386.rpm
- cross-armv4l-glibc-profile-2.2.1-2mz.i386.rpm
- cross-armv4l-jpeg-6b-2mz.i386.rpm
- cross-armv4l-jpeg-devel-6b-2mz.i386.rpm
- cross-armv4l-kernel-headers-2.4.5_rmk7_np2-1mz.i386.rpm
- cross-armv4l-libfloat-1.0-3mz.i386.rpm
- cross-armv4l-libpng-1.0.8-4mz.i386.rpm
- cross-armv4l-openobex-0.9.8-1mz.i386.rpm
- cross-armv4l-openssl-0.9.6d-1mz.i386.rpm
- cross-armv4l-openssl-devel-0.9.6d-1mz.i386.rpm
- cross-armv4l-popt-1.5-1mz.i386.rpm
- cross-armv4l-zlib-1.1.3-5mz.i386.rpm

该工具链适应于 StrongARM、xScale 等处理器，和上一个工具链的不同之处在于 libc 的版本，此外该工具链包括了许多有用的库，而且该开发工具链可以很好地和 Autoconf/Automake 脚本配合使用。

首先按如下次序安装二进制工具和编译器及 libc：

```
# rpm -ivh cross-armv4l-binutils-2.10-3mz.i386.rpm
# rpm -ivh cross-armv4l-gcc-2.95.2-10mz.i386.rpm
# rpm -ivh cross-armv4l-glibc-2.2.1-2mz.i386.rpm
# rpm -ivh cross-armv4l-glibc-profile-2.2.1-2mz.i386.rpm
```

然后安装其它需要的库。

工具链将安装到/opt/host/armv4l 目录下，安装完之后设置 PATH 环境变量:

```
$ export PATH=/opt/host/armv4l/bin:$PATH
```

然后就可以进行应用程序的交叉编译了。具体命令可参考本手册第 5 章“MiniGUI 的交叉编译”。

6.1.3 mips 交叉编译工具链

增值版提供的 mipsel 交叉编译工具链包括如下软件包：

- binutils-mipsel-linux-2.13.2.1-3.i386.rpm
- egcs-c++-mipsel-linux-1.1.2-4.i386.rpm
- egcs-libstdc++-mipsel-linux-2.9.0-4.i386.rpm
- egcs-mipsel-linux-1.1.2-4.i386.rpm
- egcs-objc-mipsel-linux-1.1.2-4.i386.rpm

首先安装如下软件包：

```
# rpm -ivh binutils-mipsel-linux-2.13.2.1-3.i386.rpm
# rpm -ivh egcs-mipsel-linux-1.1.2-4.i386.rpm
```

然后安装其它需要的软件包。

需要注意的是，该工具链中不包括 libc 库，需要另外安装。

6.1.4 ppc 交叉编译工具链

ppc-linux-toolchain.tgz 交叉编译工具链适用于 PowerPC 8xx 系列的处理器，它的安装比较简单，直接把该包解压到某个目录下，如/usr/local/ppc-linux/，然后设置 PATH 环境变量就可以了。

6.1.5 uClinux 交叉编译工具链

产品光盘中包含了两个用于 uClinux 的编译工具链，m68k-elf-tools-20030314.sh 和 arm-elf-tools-20030314.sh，分别用于 m68k 和 arm 无 MMU 平台的编译。这两个工具链均可以从http://www.uclinux.org/pub/uClinux/下载。

这两个工具链都是自解压的包，需要以 root 身份执行：

```
sh ./m68k-elf-tools-20030314.sh
sh ./arm-elf-tools-20030314.sh
```

相关的工具和文件将安装到/usr/local 目录下，m68k 和 arm 的 gcc 编译器分别为 m68k-elf-gcc 和 arm-elf-gcc。

6.2 MiniGUI 图形配置工具的使用

MiniGUI 1.3 版本中提供了图形界面的配置工具，使得 MiniGUI 的配置、编译和交叉编译更加直观和方便。

6.2.1 配置工具的运行和操作

要运行图形界面配置工具，首先进入 MiniGUI 库源代码目录，如 libminigui-1.6.x，然后在终端上输入：

```
$ make menuconfig
```

该命令将启动图形界面的配置工具。配置工具的主界面截图见图 6-1。

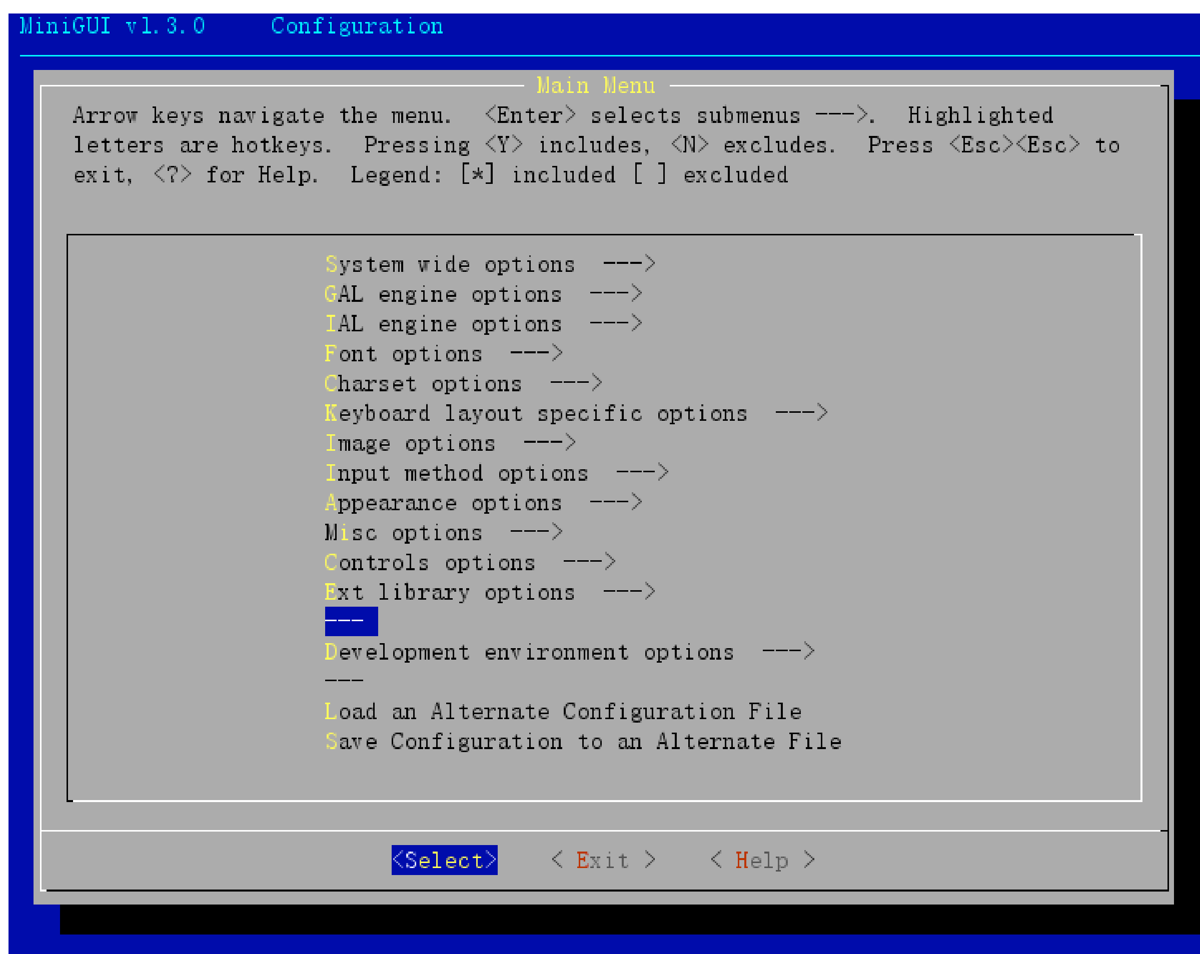


图 6-1 配置工具主界面

该图形配置界面的操作方法和 Linux 内核源代码中 make menuconfig 产生的配置界面的用法是一样。我们可以按箭头键来选择当前项，通过回车键进入子菜单，通过空格键选中或取消某个配置选项（或者按 y 和 n 键），连续按 ESC 键退出到上一个菜单。如图 6-2 所示。

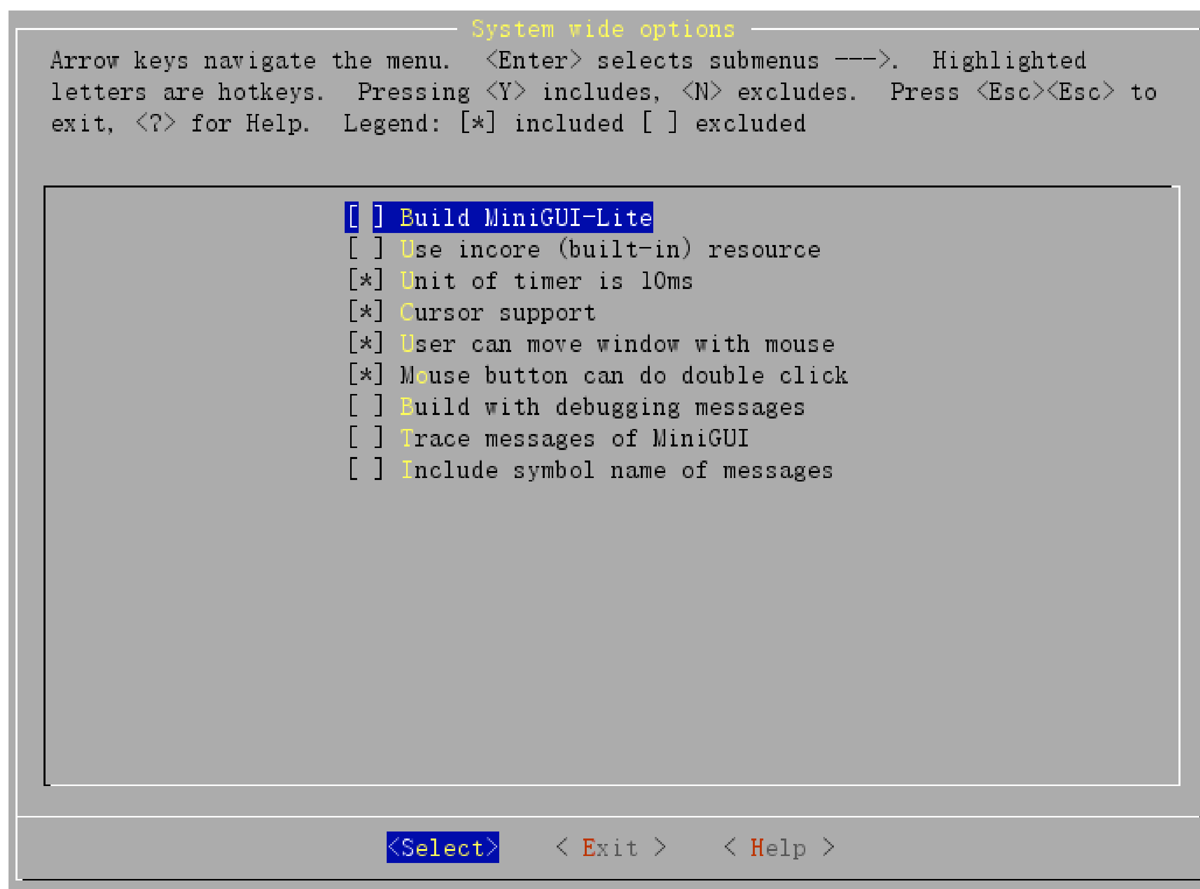


图 6-2 MiniGUI 的配置选项

在某个选项之上按？键可以查看帮助。如图 6-3 所示。



图 6-3 配置选项的帮助信息

MiniGUI 的配置选项按类别划分如下：

- 系统全局选项
- GAL 引擎选项
- IAL 引擎选项
- 字体选项
- 字符集选项
- 键盘布局选项
- 图像文件支持选项
- 输入法选项
- 外观选项
- 其它选项
- 控件
- 扩展库选项
- 开发环境设置选项

用户可以选择

```
“Save Configuration to an Alternate File”
```

把配置选项保存为配置文件，在需要的时候选择

```
“Load an Alternate Configuration File”
```

把配置文件中保存的配置选项载入。

用户运行“make menuconfig”启动配置工具时，初始的配置选项使用上次保存的配置选项；如果没有的话将使用缺省的配置文件 configs/defconfig，该配置文件中的选项设置是 MiniGUI 的默认配置。如果想要在进入图形配置界面时使用 MiniGUI 的默认配置，运行“make defconfig”。

从配置界面保存配置选项并退出之后，配置脚本将根据所设置的配置选项进行 MiniGUI 编译选项的配置。配置完成之后，用户就可以通过 make 命令来编译 MiniGUI 库，make install 来安装。

6.2.2 交叉编译器的选择

用户可以通过在开发环境菜单“Development environment options”的编译器选择子菜单中选择编译 MiniGUI 库所使用的编译器或交叉编译器。默认情况下的编译器为 i386 平台上的 gcc 编译器，如果需要对 MiniGUI 进行交叉编译的化，可以根据目标平台的类型选择适当的交叉编译器。增值版产品光盘中提供了 ARM、PowerPC、MIPS 和 M68k 平台上的一系列交叉编译工具链。如图 6-4 所示。

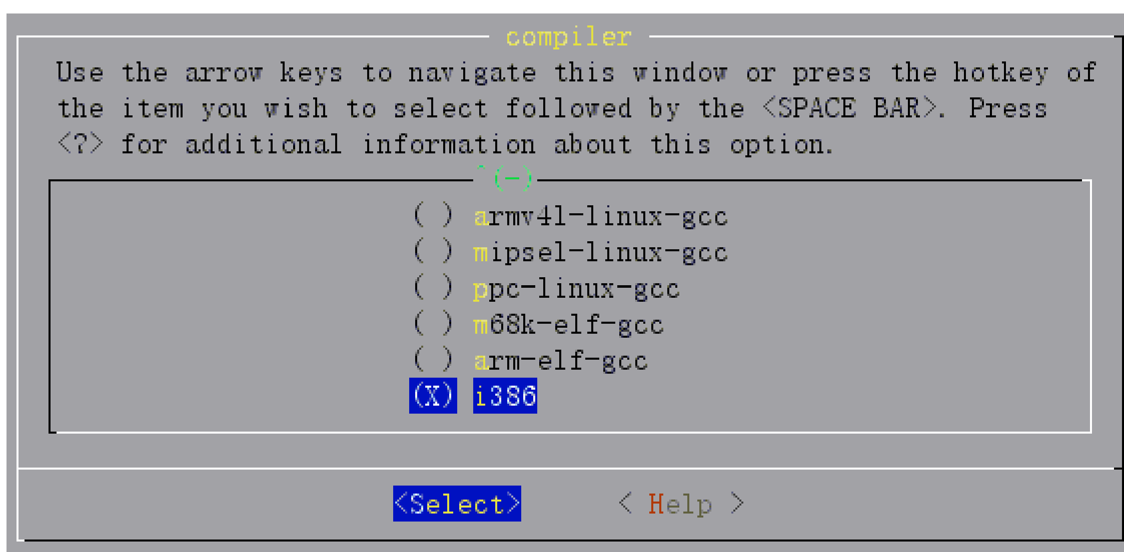


图 6-4 选择交叉编译器

【注意】必须安装增值版中提供的相应交叉编译工具之后，在选项中选择交叉编译器才有意义。

6.2.3 libc 的选择

C 库是 Linux 操作系统必需的基本库。一般而言，Linux 系统都使用 GNU 的 glibc 库，该库是一个成熟的、优秀的但是却很庞大的 C 库。

uClibc 是针对嵌入式系统而设计的轻量级 C 库，稳定、强大且具有高度兼容性，可以用作嵌入式系统上的 Glibc 替代品。在绝大多数情况下，针对 uClibc 编译的应用程序和工具与针对 glibc 编译的没有分别。uClibc 网站（www.uClibc.org）有 uClibc 的详细介绍和已经移植到 uClibc 的应用程序列表。

MiniGUI 完全可以在 uClibc 之上运行，不过由于 uClibc 尚不支持无 MMU 的 m68k 和 arm7 等平台上的函数库动态链接，所以 MiniGUI 只能编译为静态库来使用。

注意，必须先安装 uClibc 才能在上面的选项中选择 uClibc；如果您正在使用 uClinux-dist 进行 uClinux 的编译的话，就不需要另外安装 uClibc 库了。

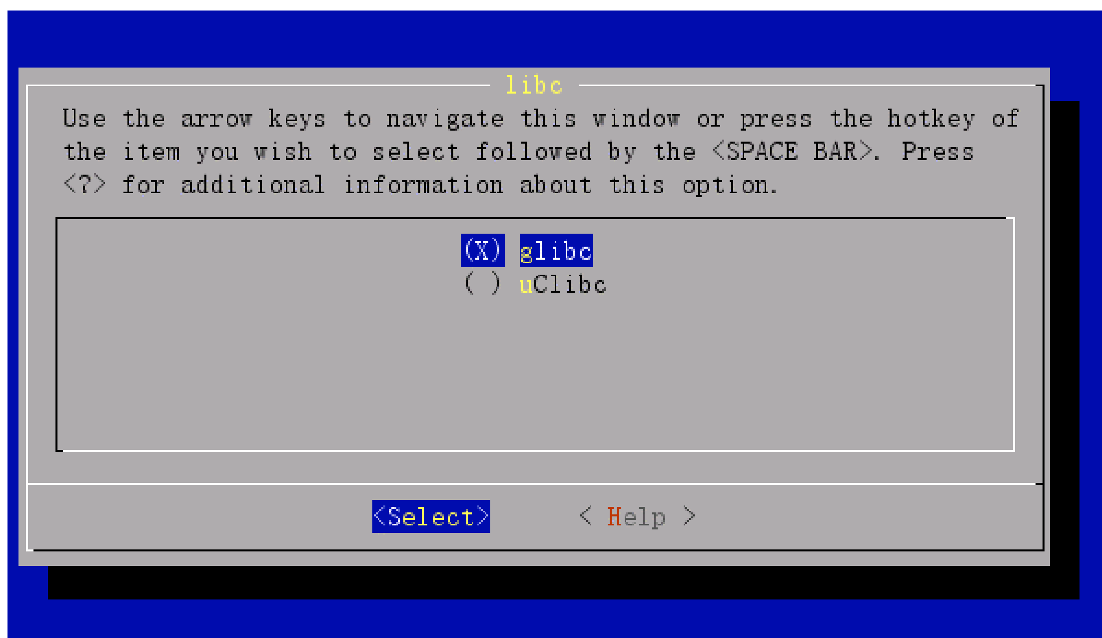


图 6-5 选择 libc

图 6-5 给出了 C 函数库的选择界面。

6.3 相关库的源代码

增值版中提供了 MiniGUI 相关依赖库的源代码，用户可以根据需要自己进行（交叉）编译并在目标系统中使用。

- libfreetye：TrueType 字体支持库
- libt1：Adobe Type1 字体支持库
- libjpeg：JPEG 图像支持库
- libpng：PNG 图像支持库

这些函数库保存在增值版产品 CD-ROM 的 deplibs/ 目录下。

7 在 uClinux 上运行 MiniGUI

uClinux（http://www.uclinux.org）操作系统是 Linux 针对没有内存管理单元（MMU）的微处理器/微控制器的衍生版本，没有 MMU 支持是 uClinux 与 Linux 的主要差异。UClinux 继承了 Linux 的绝大部分特性，Linux 上的应用程序不需要做任何改变或者做很少的改变就可以移植到 uClinux 之上。事实上，MiniGUI 1.3 之前的版本都可以在 uClinux 之上正常运行。

由于 uClinux 具有体积小、功能强和性能好等优点以及开放源码的特性，在嵌入式系统开发中得到了广泛的应用。当然，由于没有 MMU 支持，uClinux 也有一些限制，例如进程没有独立的地址空间，内存空间没有保护，对共享库的支持不够等等。

在 1.3 版本中，MiniGUI 针对 uClinux 系统资源少的特点进行了专门的优化，减少了 MiniGUI 的资源消耗，而且通过--enable-incoreres 选项可以把 MiniGUI 所需的位图等资源编译到目标文件中，不再需要单独的位图、图标和字体等资源文件以及系统配置文件 MiniGUI.cfg。

7.1 配置和编译 uClinux

首先您需要 uClinux 的交叉编译工具，产品光盘中包括了 m68k-elf 和 arm-elf 两套可用于 uClinux 的工具链，安装和使用方法见 6.1.6 “uClinux 交叉编译工具链”。

其次，您需要有一个 uClinux 发行版，产品光盘中包括了一个 uClinux 发行版 uClinux-dist-20030522，该发行版也可从 uClinux 网站下载。把包在某个目录下解开：

```
tar zxvf uClinux-dist-20030522.tar.gz
```

该命令将把 uClinux 发行版目录树解开到一个名为 uClinux-dist 的目录下。进入该源代码目录：

```
cd uClinux-dist
```

由于该发行版的 xcopilot 部分有一些 bug，所以我们需要打一个补丁：

```
patch -p1 < (patch所在目录)/uClinux-dist-20030522.patch
```

然后进行 Linux 内核和应用程序的配置：

```
make menuconfig
```

运行该命令将进入 uClinux-dist 的图形配置界面，如图 7-1 所示。

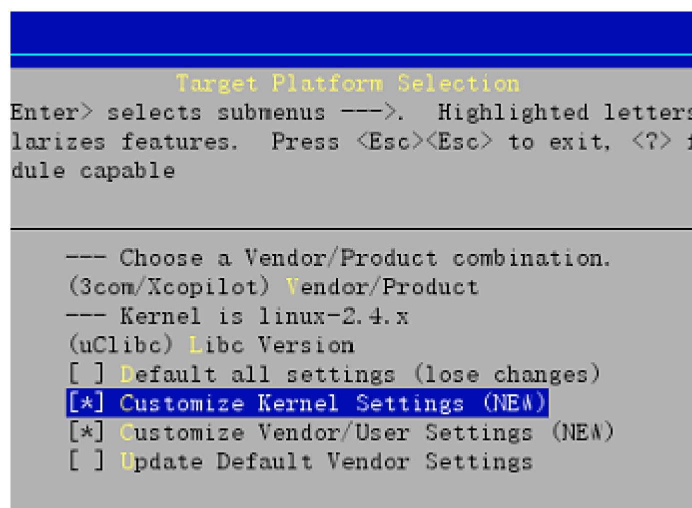


图 7-1 uClinux 配置界面

我们以 Xcopilot 模拟器上的 uClinux 和 MiniGUI 编译为例，所以这里我们在“Vendor/Product”选项中选择目标平台为 3com/Xcopilot。Kernel 选为 Linux-2.4.x，libc 选 uClibc。然后选中“Customize Kernel Settings”选项，进行内核的配置。

进入内核配置的“Character devices”菜单，如图 7-2 所示。选中“68328 digitizer support”和“Virtual terminal”。“68328 digitizer support”是 xcopilot 的触摸屏支持，“Virtual terminal”是虚拟终端支持，选中该项才能使“Frame-buffer support”选项出现。

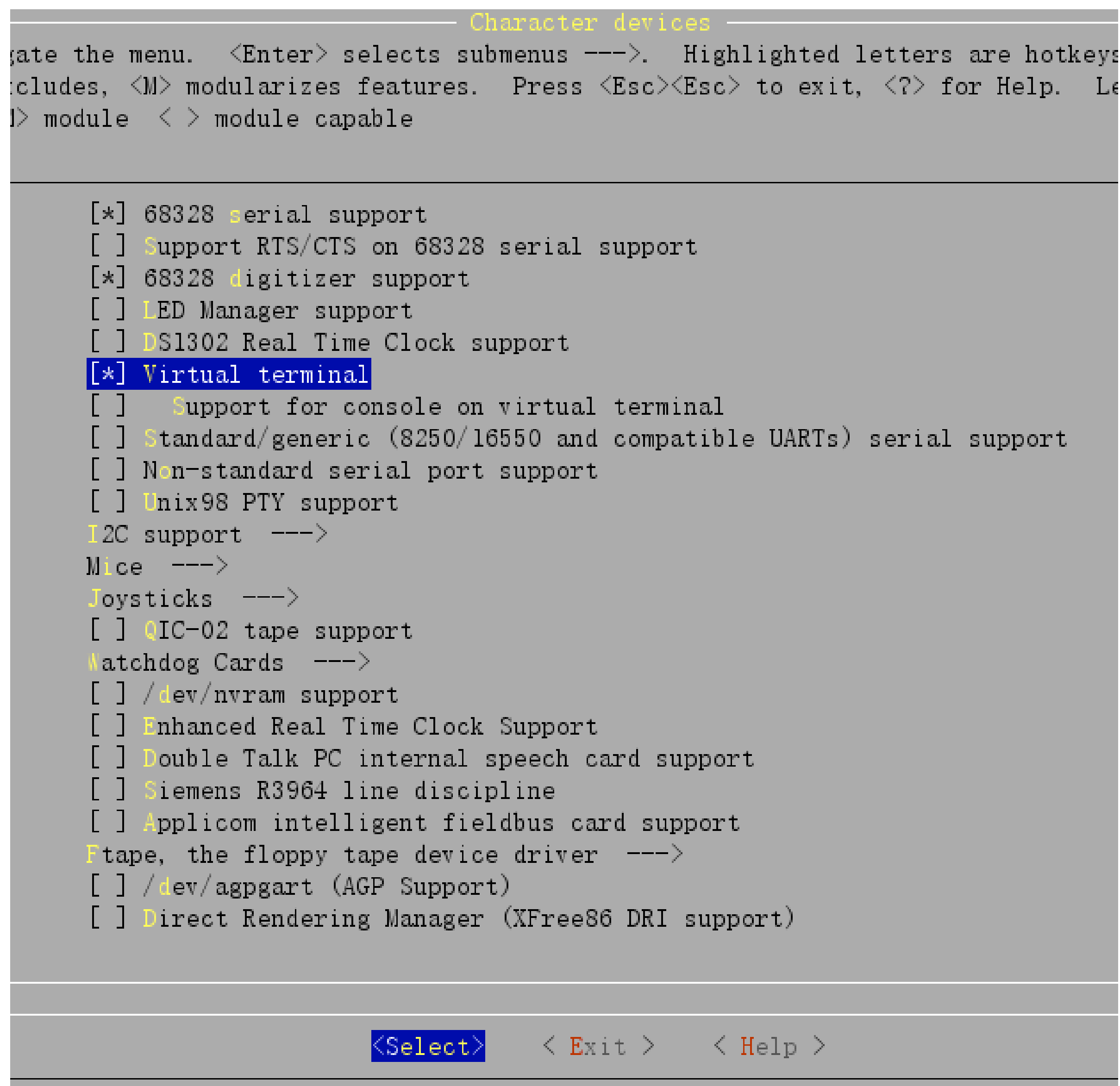


图 7-2　字符设备配置选项

进入“Frame-buffer support”子菜单，选中“Support for frame buffer devices”，然后选择“Dragonball frame buffer”支持 xcopilot 的 framebuffer，选中“Advanced low level driver options”，然后选择“Monochrome support”，因为 xcopilot 是单色显示。如图 7-3 所示。

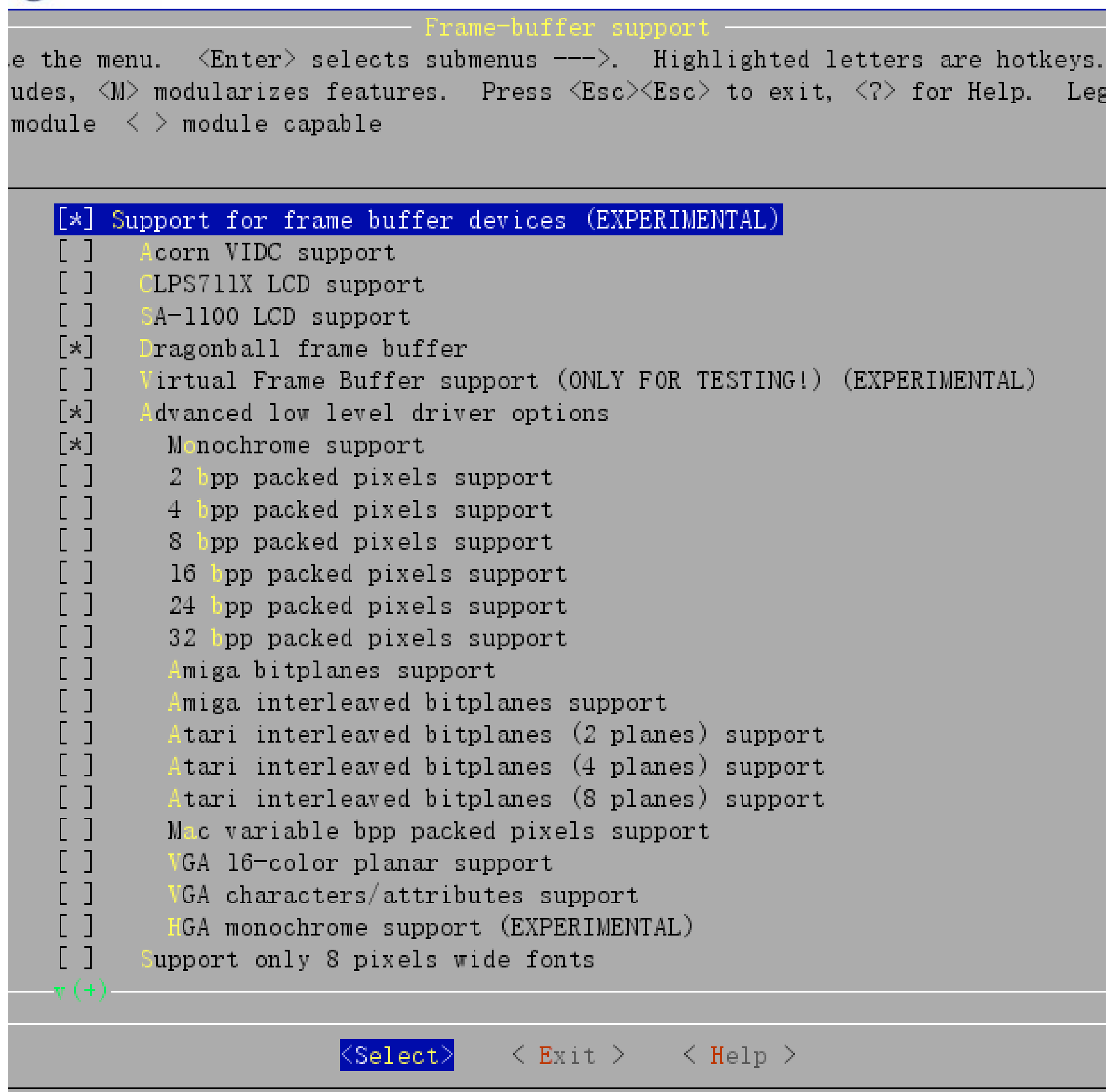


图 7-3 FrameBuffer 配置选项

以上是和 MiniGUI 相关的内核配置选项。

7.2 配置和编译 uClibc

进入 uClinux-dist/uClibc 目录，运行

```
make TARGET_ARCH=m68k menuconfig
```

进入 uClibc 的图形配置界面(如果是别的平台，把 m68k 改为相应的平台名称)。如图 7-4 所示。

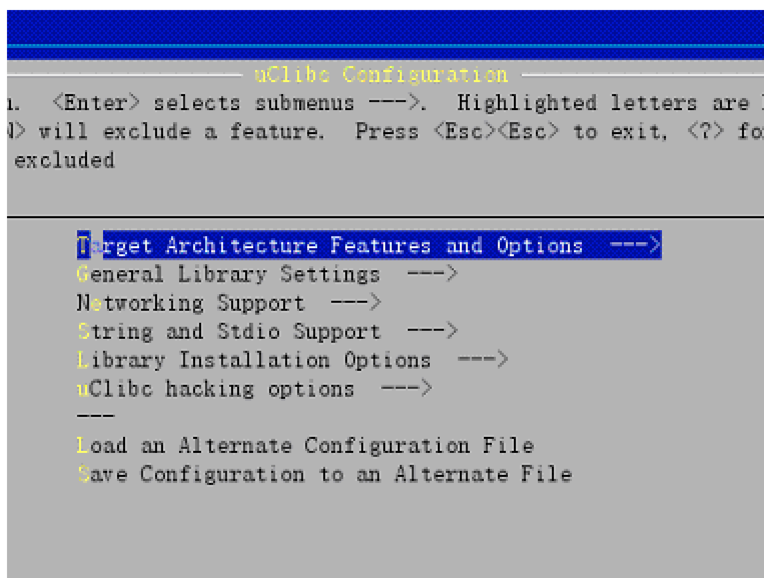


图 7-4 uClibc 配置界面

进入“Target Architecture Features and Options”子菜单，选中“Enable floating point number support” 和 “Enable full C99 math library support”。如图 7-5 所示。

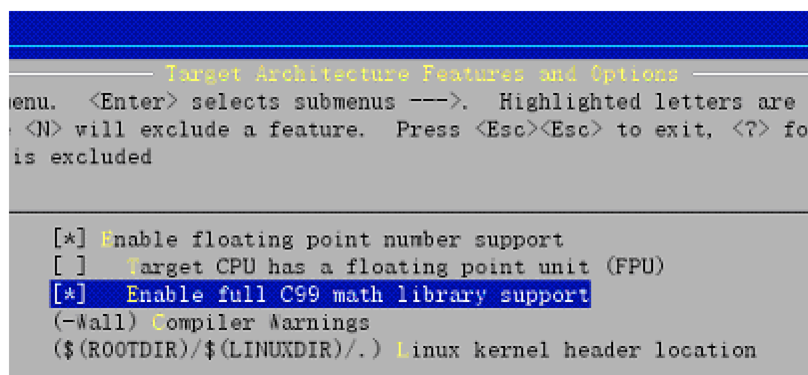


图 7-5 浮点和数学库支持

然后进入“General Library Settings”子菜单，如果 MiniGUI 需要编译为线程版的话，选择“POSIX Threading Support”线程支持。如图 7-6 所示。

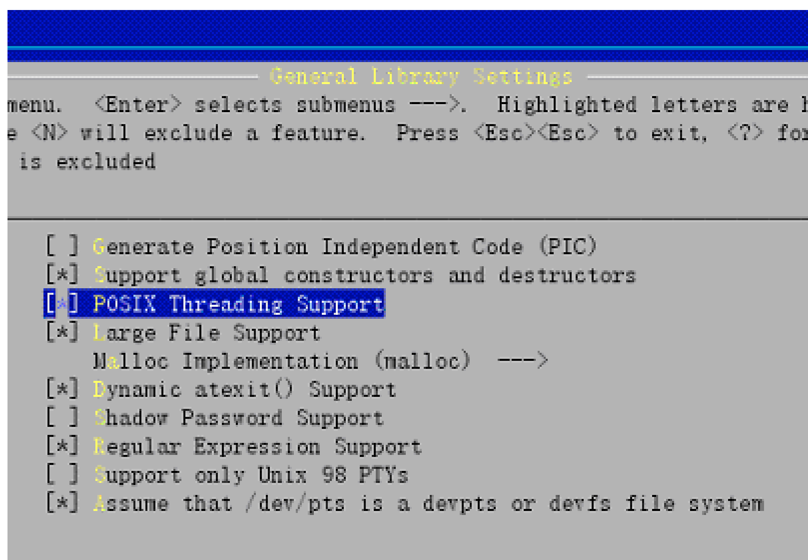


图 7-6 POSIX 线程支持

配置完 uClinux 和 uClibc 之后，在 uClinux-dist 目录下：

```
make dep
make
```

就可以对内核和 C 库及应用程序进行编译，如果成功的话将生成相应目标平台的映像文件，对于 xcopilot 来说就是 images 目录下的 pilot.rom 文件。

7.3 配置和编译 MiniGUI

下面介绍如何使用图形界面配置工具来配置和编译用于 uClinux 的 MiniGUI。

进入 MiniGUI 源代码目录：

```
cd libminigui-1.6.x
```

进入图形配置界面：

```
make menuconfig
```

和 uClinux 下的配置密切相关的是“Development environment options”菜单下的选项。如图 7-7 所示。

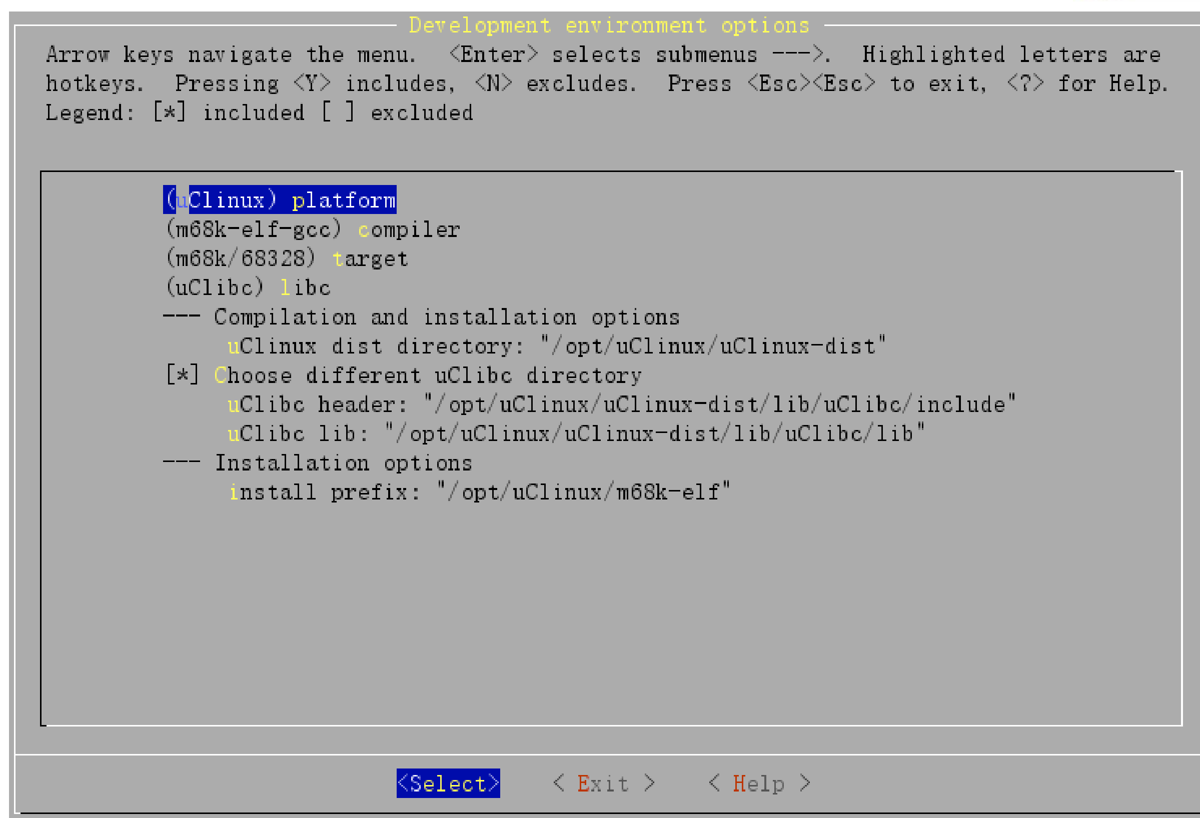


图 7-7 开发选项菜单

进入该菜单，首先要进入“platform”单选子菜单选择 MiniGUI 运行的操作系统平台，目前是 Linux 或者 uClinux，当然这里我们选择 uClinux。如图 7-8 所示。

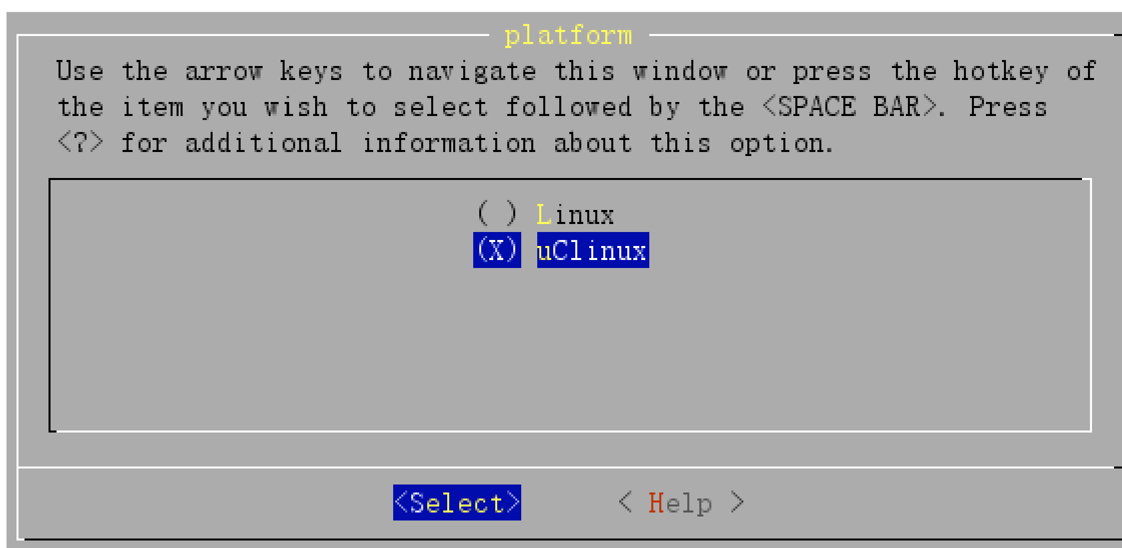


图 7-8 选择操作系统

之后我们要选择交叉编译器，在 uClinux 下目前是 m68k-elf-gcc 或者 arm-elf-gcc，分别用于 m68k-nommu 平台和 arm-nommu 平台的交叉编译。如图 7-9 所示。

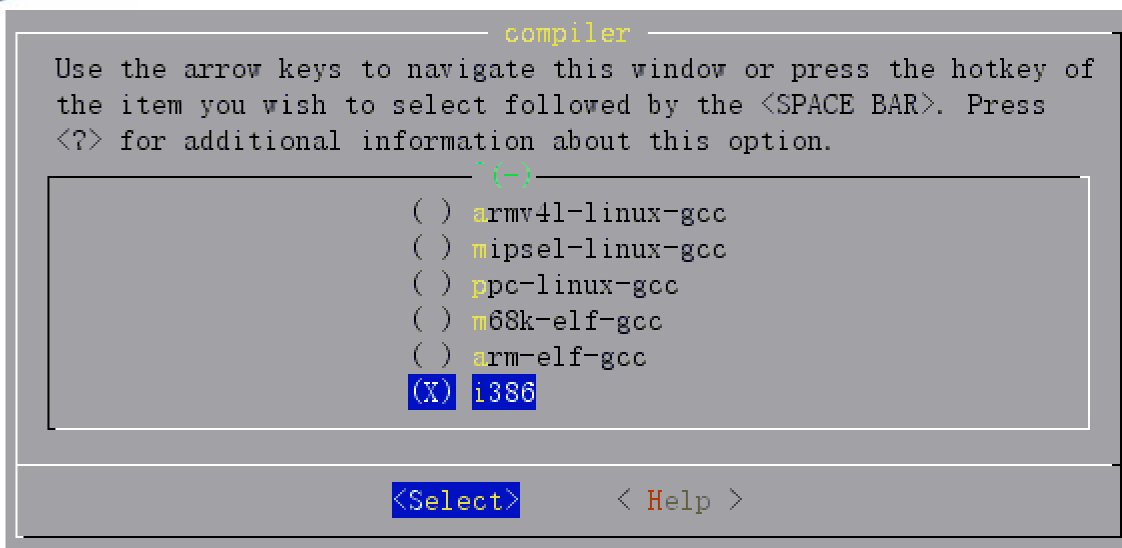


图 7-9 选择交叉编译器

在确定编译器之后，我们可能需要确定目标处理器的类型（如图 7-10 所示），包括：

- mc68328
- mc68ez328
- coldfire 5200
- coldfire 5307

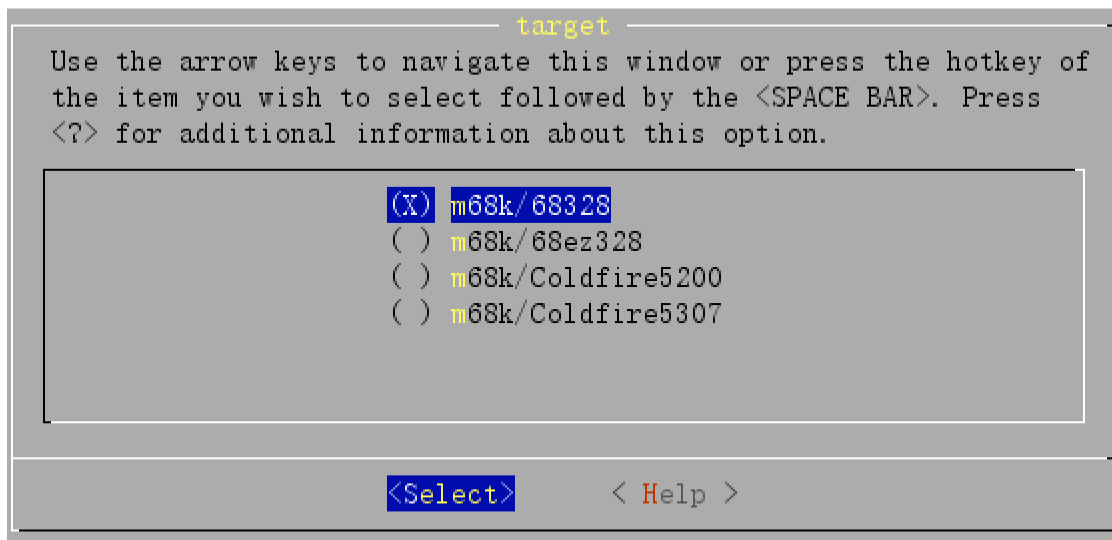


图 7-10 选择目标处理器

uClinux 下既可以用 glibc，也可以用 uClibc，一般情况下用比较小的 uClibc。进入 libc 选项选择您使用的 C 库。

我们还需要确定 uClinux-dist 所在的目录，编译 MiniGUI 时需要用到 Linux kernel 头文件和 uClibc 头文件，以及包括 C 库在内的链接库。在“uClinux dist directory”选项上

按下回车键将进入 uClinux-dist 目录设置输入框，填上您的 uClinux-dist 路径。如图 7-11 所示。

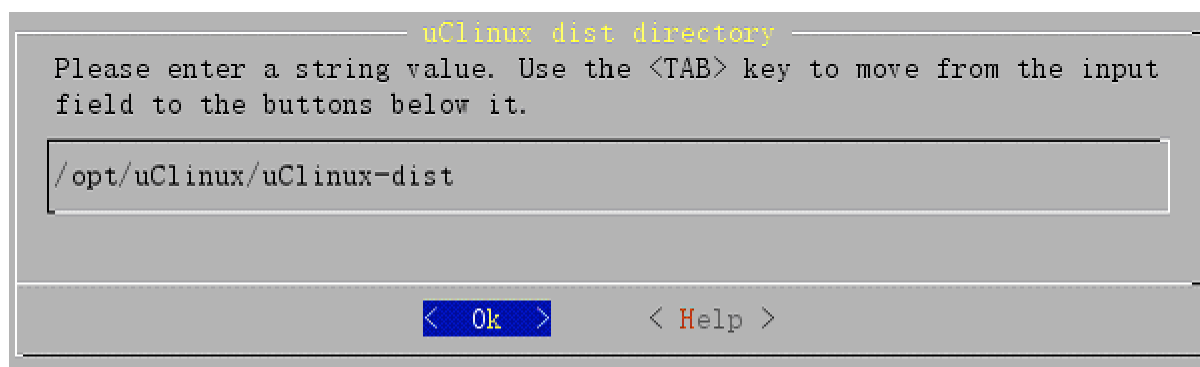


图 7-11　设置 uClinux 发行版路径

如果不使用 uClinux-dist 中的 uClibc，而是采用另外的 uClibc 头文件和库的话，可以选中“Choose different uClibc directory”，然后分别设置相应的 uClibc 头文件路径和库路径。

7.4 编译 MiniGUI 应用程序

编译 MiniGUI 应用程序有两种办法：第一种办法是把 MiniGUI 应用程序添加到 uClinux-dist 的 user 目录中，在 uClinux-dist 中进行编译和链接，直至生成最后的映像文件；另一种办法是先单独编译 MiniGUI 程序，生成可执行程序文件，然后放到 uClinux-dist 的 romfs 目录中，执行 make image 命令生成映像。

产品光盘中包括了使用第一种方法的例子：mgdemo-uclinux.tar.gz。

在 uClinux-dist/user 目录下解开该包：

```
tar zxvf mgdemo-uclinux.tar.gz
```

将在 user 目录下生成一个 mgdemo 目录，该目录包括了三个 MiniGUI 示例程序的源代码代码，分别是 bomb，ctrldemo 和 gdidemo。

修改 mgdemo 下的 Makefile 文件，把下列两行所指定的 minigui 头文件和库目录改为您安装的交叉编译的 MiniGUI 所在的目录：

```
CFLAGS+= -I/opt/uClinux/m68k-elf/include/
LDFLAGS+= -L/opt/uClinux/m68k-elf/lib
```

如果您安装 MiniGUI 时 prefix 为/opt/uClinux/m68k-elf 的话就不用修改了。

修改 user 目录下的 Makefile 文件，加上一行：

```
dir_y += mgdemo
```

这样在 uClinux-dist 目录下输入“make”进行编译时就把该示例程序包括在编译范围之内，编译后将在 romfs/bin 目录下生成三个程序文件：bomb，ctrldemo 和 gdidemo。

7.5 xcopilot 模拟器的使用

把 xcopilot 模拟器源代码包解开：

```
tar zxvt xcopilot-0.6.6-uc0.tar.gz
```

编译 xcopilot：

```
cd xcopilot-0.6.6-uc0
./configure
make
ln -s {你的uClinux-dist目录}/images/pilot.rom ~/.xcopilot/pilot.rom
```

运行：

```
cd xcopilot-0.6.6-uc0
./xcopilot
```

图 7-12 和图 7-13 是 MiniGUI 演示程序在 Xcopilot 模拟器上的运行截图。



图 7-12 运行 uClinux 和 MiniGUI 的 xcopilot 模拟器



图 7-13 xcopilot 上的 MiniGUI 月历控件

附录 A MiniGUI 的三种运行模式

A.1 MiniGUI 三种运行模式的来由

MiniGUI 最初的版本，即 MiniGUI-Threads 采用了线程机制（类似 Windows CE），这样，所有的应用程序都运行在同一个地址空间，从而大大提高了程序之间的通讯效率。但显然，这种基于线程的结构也导致了系统整体的脆弱——如果某个线程因为非法的数据访问而终止运行，则整个进程都将受到影响。不过，这种体系结构对实时控制系统等时间关键的系统来讲还是非常适合的。

为了解决 MiniGUI 版本因为线程而引入的一些问题，同时也为了让 MiniGUI 更加适合一些需要高度可扩展性的复杂嵌入式系统，我们决定开发 MiniGUI-Lite 版本。这个版本的开发目的是：

1) 保持与原先 MiniGUI 版本在源代码级 99% 以上的兼容。
2) 不再使用线程库。
3) 可以同时运行多个基于 MiniGUI-Lite 的应用程序，即多个进程，并且提供前后台进程的切换。

显然，要同时满足上述三个目的，如果采用传统的客户/服务器（C/S）结构对现有 MiniGUI 进行改造，应该不难实现。但前面提到的传统 C/S 结构的缺陷却无法避免。经过对 PDA 等嵌入式系统的分析，我们发现，某些 PDA 产品具有运行多个任务的能力，但同一时刻在屏幕上进行绘制的程序，一般不会超过两个。因此，只要确保将这两个进程的绘制相互隔离，就不需要采用复杂的 C/S 结构处理多个进程窗口之间的互相剪切。也就是说，在这种产品中，如果采用基于传统 C/S 结构的多窗口系统，实际是一种浪费。

有了上述认识，我们对 MiniGUI 版本进行了如下简化设计：

- 每个进程维护自己的主窗口 Z 序，同一进程创建的主窗口之间互相剪切。也就是说，除这个进程只有一个线程，只有一个消息循环之外，它与原有的 MiniGUI 版本之间没有任何区别。每个进程在进行屏幕绘制时，不需要考虑其他进程。
- 建立一个简单的客户/服务器体系，但确保最小化进程间的数据复制功能。因此，在服务器和客户之间传递的数据仅限于输入设备的输入数据，以及客户和服务器之间的某些请求和响应数据。
- 有一个服务器进程（mginit），它负责初始化一些输入设备，并且通过 UNIX Domain 套接字将输入设备的消息发送到前台的 MiniGUI-Lite 客户进程。
- 服务器和客户被分别限定在屏幕的某两个不相交矩形内进行绘制，同一时刻，只能

有一个客户及服务器进行屏幕绘制。其他客户可继续运行，但屏幕输入被屏蔽。服务器可以利用 API 接口将某个客户切换到前台。同时，服务器和客户之间采用信号和 System V 信号量进行同步。

- 服务器还采用 System V IPC 机制提供一些资源的共享，包括位图、图标、鼠标、字体等等，以便减少实际内存的消耗。

从传统 C/S 窗口系统的角度看，MiniGUI-Lite 的这种设计，无法处理达到完整的多窗口支持，这的确是一个结构设计上的不足和缺陷。不过，这实际是 MiniGUI-Lite 不同于其他窗口系统的一个特征。因为处理每个进程之间的互相剪切问题，将导致客户和服务器之间的通讯量大大增加，但实际上在许多嵌入式系统当中这种处理是没有必要的。在类似 PDA 的嵌入式系统中，往往各个程序启动后，就独占屏幕进行绘制输出，其他程序根本就没有必要知道它现在的窗口被别的进程剪切了，因为它根本就没有机会输出到屏幕上。所以，在 MiniGUI-Lite 当中，当一个进程成为最顶层程序时，服务器会保证其输出正常，而当有新的程序成为最顶层程序时，服务器也会保证其他程序不能输出到屏幕上。但这些进程依然在正常执行着，不过，服务器只向最顶层的程序发送外部事件消息。

此外，我们还可以通过--enable-standalone 配置选项把 MiniGUI 配置为单独运行的进程版。在 standalone 版本中，一个时刻只能有一个应用程序在运行。standalone 版中的应用程序其实就是 Lite 版中的服务器程序，不过在 standalone 版中它的名字不限定为 mginit。standalone 版适合简单的应用场合，一般情况下推荐使用 MiniGUI-Threads。

A.2 MiniGUI 三种运行模式的使用场合

MiniGUI 的三种运行模式分别适合不同的应用场合。用户应根据具体的系统需求来选择特定的运行模式：

- MiniGUI-Threads。运行在 MiniGUI-Threads 上的程序可以在不同的线程中建立多个窗口，但所有的窗口在一个进程或者地址空间中运行。这种运行模式非常适合于大多数传统意义上的嵌入式操作系统，比如 uC/OS-II、eCos、VxWorks、pSOS 等等。当然，在 Linux 和 uClinux 上，MiniGUI 也能以 MiniGUI-Threads 的模式运行。因此，我们在 uC/OS-II、eCos、VxWorks 等操作系统上运行 MiniGUI 时，需要将 MiniGUI 配置成 MiniGUI-Threads 模式。在嵌入式 Linux 操作系统中，如果应用程序的功能单一，则可以选择 MiniGUI-Threads 模式。

- MiniGUI-Lite。和 MiniGUI-Threads 相反，MiniGUI-Lite 上的每个程序是单独的进程，每个进程也可以建立多个窗口。MiniGUI-Lite 适合于具有完整 UNIX 特性的嵌入式操作系统，比如嵌入式 Linux。如果应用程序的功能复杂，而且需要某种

程度上的健壮性和扩展性的话，则选择 MiniGUI-Lite 模式。目前，MiniGUI-Lite 模式仅支持功能完整的嵌入式 Linux 操作系统。

- MiniGUI-Standalone。这种运行模式下，MiniGUI 可以以独立进程的方式运行，既不需要多线程也不需要多进程的支持，这种运行模式适合功能单一的应用场合。比如在一些使用 uClinux 的嵌入式产品中，因为各种原因而缺少线程支持，这时，就可以使用 MiniGUI-Standalone 来开发应用软件。MiniGUI-Standalone 模式可适合于所有的嵌入式操作系统。

附录 B 有关 LibGGI 和 SVGALib

这两个函数库可以为 MiniGUI 提供底层图形支持，我们称之为“图形引擎”。其中 SVGALib 是一个比较老的函数库，只提供对 Linux 控制台的支持；LibGGI 是一个比较新的图形函数库，提供了对 Linux 控制台、X Window 等的支持，并且接口相对简单。这两个函数库在开发 MiniGUI 的早期阶段具有重要地位，但现在，因为 Linux 内核中具有抽象的 FrameBuffer 设备支持，这两个图形引擎基本上失去了实用价值。

因此，在支持 FrameBuffer 的 Linux 平台上（使用 2.2 以上的 Linux 内核），就没有任何必要使用这两个函数库来为 MiniGUI 提供图形引擎支持，您只要使用 MiniGUI 提供的 NATIVE 或 FBCON 引擎就能正常运行 MiniGUI 应用程序。另外，SVGALib 和 LibGGI 这两个图形引擎只能用来支持 MiniGUI-Threads，而不能用来支持 MiniGUI-Lite。因此，如果没有特别需求，就无须考虑使用这两个图形引擎。

如果读者使用的是 Linux 内核 2.0.xx，就必须使用 SVGALib 或者 LibGGI。SVGALib 函数库实际由两个库组成，即 vga 和 vgagl。MiniGUI 使用的是 vgagl，该函数库在 vga 之上运行，提供了较好的图形函数。需要注意的是，MiniGUI 的运行需要 SVGALib 1.4.3 及以上的版本支持。

LibGGI 比起 SVGALib 好的地方在于，使用 LibGGI 的应用程序可以在 X Window 上运行，而无须重新编译。

以上提到的函数库可以从飞漫软件的主页上下载：

```
http://www.minigui.com/download/c3rdparty.sthml。
```

也可以访问上述两个自由软件项目的站点：

```
http://www.ggi-project.org
http://www.svgalib.org
```

除 SVGALib 之外，LibGGI 提供有 automake/autoconf 接口，从而可以非常方便地编译并安装这些依赖库。LibGGI 还提供了一个批处理脚本，可以帮助您编译并安装 LibGGI 的所有函数库，其中包括用来处理输入（键盘、鼠标等）的 libgii，用来处理图形输出的 libggi 等。对 MiniGUI 来说，只需安装 libgii 和 libggi。在下载 libggi-2.1beta2.0-20000316.tgz 文件之后，运行

```
$ tar zxf libggi-2.1beta2.0-20000316.tgz
```

可解开该 tgz 包，并在当前目录中建立 degas 子目录。进入 degas/lib/ 目录，运

行 ./buildall 脚本，就可以编译并安装 libgii、libggi 等函数库。注意应该以 root 身份运行 buildall 脚本。如果要单独编译某个函数库，可以进入相应的目录，然后按照正常的编译过程进行编译。比如，要重新编译 libggi，可运行如下命令：

```
$ cd libggi
$ ./autogen.sh
$ ./configure
$ make
$ su -c  make install
```

注意在编译 libggi 之前，首先要编译并安装 libgii 库。对 SVGALib 来说，只要解开软件包并运行 make 就可以编译并安装函数库。注意在安装完成每个函数库之后，要运行 ldconfig 命令更新共享库的搜索缓存，并确保将 /usr/local/lib 目录添加到 /etc/ld.so.conf 文件。

需要说明的是，为支持 MiniGUI 在具有较低配置的 PC 机上运行，我们还提供了对 VGA 标准 16 色显示模式的支持。这个引擎称为 VGA16，也是建立在 SVGALib 之上的。不过，这个引擎可支持 MiniGUI-Threads 和 MiniGUI-Lite 两个版本。要使用 VGA16 引擎，应该在配置 MiniGUI 时使用如下选项：

```
--disable-newgal --enable-vga16gal
```

并在 MiniGUI.cfg 中指定 gal_engine=vga16。

附录 C 建立 MiniGUI 的 PC 运行环境

C.1 MiniGUI 在字符控制台上的运行：配置 FrameBuffer

要想使用 FrameBuffer 环境，需要做两个工作：

- 一个支持 VESA FrameBuffer 的内核（2.2 以上的版本即可）。
- 配置 LILO 或者 GRUB 的启动选项，使内核启动时能切换到指定的显示模式，如果已经有了一个支持 FrameBuffer 的内核，可以跳过第一步。

通常，通过运行 fbset 命令可以了解内核是否支持 FrameBuffer。若系统中没有安装 fbset 命令，可通过下面的方法确认是否已包含了 FrameBuffer 驱动程序：

- 查看 /proc/devices，观察是否有 fb 设备；
- 运行 cat /dev/fb0 > /dev/null 命令，观察是否能够正常打开 /dev/fb0 设备。

C.1.1 编译安装支持 FrameBuffer 的内核

首先获得一份内核的源代码，如果用户的 Linux 发行版是 Kernel 2.2 以上，就需要到 /usr/src/linux 去重新编译内核。如果您系统的内核版本比较低，请在 RedHat6.2 及以上版本 Linux 的安装盘中寻找，一般在 RPMS 里）。对 PC 机而言，一般的显示芯片是 VESA 兼容的，这时，就可以使用 VESA FrameBuffer 驱动程序。下面的示例说明了 VESA FrameBuffer 驱动程序的配置和激活方法。

1) 配置内核编译选项

```
# cd /usr/src/linux
# make menuconfig（命令行下），或xconfig（XWindow下）。
```

与 frame buffer device 有关的选项有（用空格键来进行选中或去处，其余编译选项请参考其它资料）：

```
    Code maturity level opetions
     [*] Prompt for development and/or incomplete codes/drivers
     Console drivers
     [*] Video mode selection support
     ...
     [*] Support for frame buffer devices
     ...
     [*] VESA VGA graphics console
     ...
     [*] Advance low level driver options
     ...

[Exit]
[Exit]
```

```
Do you wish to save your new kernel configuration?
[Yes]
```

2) 编译安装内核

```
# make dep
# make bzImage
# make modules
# make modules_install
```

至此，新的内核已经编译完成。然后使用新的内核启动。

把编译好的内核拷到 /boot 目录，文件名可自定，如：

```
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-2.2.16-fb
```

C.1.2 设置 LILO 的启动选项

下面是典型的 /etc/lilo.conf 文件

```
boot = /dev/hda2
timeout = 50
prompt
read-only

image = /boot/vmlinuz-2.2.16-22
    label = linux
    root = /dev/hda2
    other = /dev/hda1

image = /boot/vmlinuz-2.2.16-fb （新编译的支持 VESA FrameBuffer 内核）
    label = linuxfb （启动标号，可自定）
    root = /dev/hda2 （您的根文件系统，具体会有不同）
    vga = 0x317 （VESA 显示模式号，参照下表）
    other = /dev/hda1
```

保存所作的修改，更新启动程序。运行

```
# lilo
```

使 LILO 选项生效。

注意上面添加的 vga=0x0317 选项。该选项指定内核在启动时将显示模式设置为 1024x768 16 位色。如果您的显示芯片或者显示器无法达到这个显示模式，则可以参照下表设置一个合理的值：

```
Colours   640x400 640x480 800x600 1024x768 1280x1024 1600x1200
  --------+-----------------------------------------------------
   4 bits |    ?       ?     0x302      ?        ?         ?
   8 bits |  0x300   0x301   0x303    0x305    0x307     0x31C
  15 bits |    ?     0x310   0x313    0x316    0x319     0x31D
  16 bits |    ?     0x311   0x314    0x317    0x31A     0x31E
  24 bits |    ?     0x312   0x315    0x318    0x31B     0x31F
  32 bits |    ?       ?       ?        ?        ?         ?
```

C.1.3 设置 GRUB 的启动选项

如果您的系统使用 GRUB 内核装载器（引导器），则应该编辑 /boot/grub/menu.lst 文件，复制某个已有的内核启动选项，并在 kernel 打头的一行末尾添加 vga=0x0317 选项，如下所示：

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,0)
#          kernel /boot/vmlinuz-version ro root=/dev/hda1
#          initrd /boot/initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.4.18-3, FrameBuffer)
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hda1 vga=0x0317
        initrd /boot/initrd-2.4.18-3.img

title Red Hat Linux (2.4.18-3)
        root (hd0,0)
        kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hda1
        initrd /boot/initrd-2.4.18-3.img
```

使用 GRUB，无须像 LILO 那样在重新启动之前运行某个程序使选项生效。

C.1.4 重新启动并激活 VESA FrameBuffer 驱动程序

重启，出现 lilo: 提示符或者 GRUB 的启动画面时选择对应的支持 FrameBuffer 的选项。如果一切正常，则 Linux 引导时，会在屏幕的左上方显示一个可爱的小企鹅或者发行版厂商的 LOGO 图片。

C.1.5 使用其它的 FrameBuffer 驱动程序

如果您的显示芯片恰好是 Linux 内核的 FrameBuffer 所支持的，比如，ATI Mach64、ATI Radeon、NeoMagic、3Dfx Banshee/Voodoo3 等（详细列表可参见 Linux 内核的配置文件），则可以将对应的 FrameBuffer 驱动程序编译到内核，或者编译成模块，然后使用 fbset 程序，就可以灵活改变显示芯片所支持的显示模式。其他关于 Linux FrameBuffer 的信息，可参阅 Framebuffer-HOWTO 文档。

C.2 MiniGUI 在 X Window 上的运行：运行 QVFB

C.2.1 安装 QVFB

QVFB 是 Qt 提供的一个虚拟 FrameBuffer 工具。这个程序基于 Qt 开发，运行在 X Window 上。您可以在 Qt 2 或者 Qt 3 源代码的 src/tools 目录中找到这个程序。编译 Qt 时，会自动编译好这个程序。将程序复制到 /usr/bin 目录下，就可以使用这个程序了。

用户也可以从飞漫软件的“免费下载”区下载由飞漫打包的 QVFB 包：

```
http://www.minigui.com/download/c3rdparty.shtml
```

您可以下载这个软件包，并单独编译和安装 qvfb 程序。

C.2.2 运行并设置 QVFB

1） 在 X Window 中，打开一个终端仿真程序， 执行 qvfb & 命令。

2）点击 qvfb 的菜单：file->configure，设置成 640×480 16 位色的显示模式。

3）合理设置 /usr/local/etc/MiniGUI.cfg，如下所示：

```
[system]
gal_engine=qvfb
ial_engine=qvfb

[qvfb]
defaultmode=640x480-16bpp
display=0
```

之后，您可以就运行 MiniGUI 应用程序了。您会发现，MiniGUI 的窗口将显示在 qvfb 程序创建的窗口中。

附录 D I810 显示芯片的支持

i810（包括 i815 等系列）芯片是 Intel 提供的一种廉价显示芯片，可集成在主板上，主要用于低端 PC 台式机或者笔记本上。该芯片为了降低成本，取消了芯片的 VESA 兼容能力，因此无法通过 Linux 的 VESA FrameBuffer 驱动程序提供对这种芯片的 FrameBuffer 驱动。和这种芯片类似的还有 i830、i845 系列等 Intel 的显示芯片。不过，已经有人编写了针对 i810 芯片的 FrameBuffer 驱动程序，且已添加到正式的 Linux 内核 2.6.x 版本中。因此，要在 PC 上增加 i810 的 FrameBuffer 支持，要么使用 Linux 2.6.x 的内核，要么采用如下方法修改 2.4.x 的内核源代码。修改 2.4.x 内核以支持 i810 芯片的步骤如下：

1) 获取 I810 显示芯片的 FrameBuffer 驱动程序，可从下面的 URL 下载：

```
http://www.visuelle-maschinen.de/ctfb/i810/
```

2) 将压缩包解开，并将文件 i810fb.c i810fb.h 复制到目录：

```
/usr/src/linux/drivers/video
```

3) 修改文件：

```
/usr/src/linux/drivers/video/fbmem.c
```

在下面语句的前面：

```
static struct {
        const char *name;
        int (*init)(void);
        int (*setup)(char*);
} fb_drivers[] __initdata = {
...
```

增加如下声明：

```
#ifdef CONFIG_FB_I810
extern int i810fb_init (void) ;
extern int i810fb_setup (char*);
#endif
```

并在下面语句的前面：

```
#ifdef CONFIG_FB_VESA
        { "vesa", vesafb_init, vesafb_setup },
#endif
...
```

增加如下 I810FrameBuffer 驱动的入口：

```
#ifdef CONFIG_FB_I810
        { "i810fb", i810fb_init, i810fb_setup },
#endif
```

4) 如果用户将 I810 驱动编译成内核的一部分，需要修改文件：

```
/usr/src/linux/drivers/char/agp/agpgart_be.c
```

将下面的语句：

```
static int __init agp_init(void)
{
        init ret_val;
...
```

修改为：

```
int __init agp_init(void)
{
        static int ret_val = 0;
        static int woman = 0;
        if (woman) return ret_val;
        woman = 1;
...
```

5) 编辑目录 /usr/src/linux/drivers/video 下面的 Makefile 文件，增加一行：

```
obj-$(CONFIG_FB_I810)    += i810fb.o
```

6) 编辑目录 /usr/src/linux/drivers/video 下面的 Config.in 文件，在行：

```
bool 'Support for frame buffer devices (EXPERIMENTAL)' CONFIG_FB_RIVA
if [ "$CONFIG_FB" = "y" ]; then
        define_bool CONFIG_DUMMY_CONSOLE y
        if [ "$CONFIG_EXPERIMENTAL" = "y" ]; then
                if [ " $CONFIG_PCI" = "y" ]; then
                        tristate ' nVidia Riva support (EXPERIMENTAL)' CONFIG_FB_RIV
A
                fi
```

下面增加：

```
 if [ "$CONFIG_PCI" = "y" ]; then
     bool ' i810 support' CONFIG_FB_I810
 fi
```

7) 执行 make menuconfig，确保选中选项：

```
          Character devices --->
                 /dev/agpgart (AGP Support)
                [*]     Intel I810/I815 (on-board) support
          Console drivers --->
                Frame-buffer support --->
                        [*]  i810 support
```

8) 编译内核：

```
make dep
make bzImage
```

9) 将新编译的内核 bzImage 复制到目录/boot 下，改名为 mybzImage：

```
# cp bzImage /boot/mybzImage
```

10) 修改 /etc/lilo.conf 文件，增加：

```
image = /boot/mybzImage
append = "video=i810fb:x:1024,xv:1024,y:768,yv:768, \
        depth:16,pclk:15384,le:168,ri:8,up:29,lo:3, \
        hs:144,vs:6,sync=3,vmode:0,accel:0,font:VGA8x8"
initrd = /boot/initrd
label = i810fb
```

11) 运行 lilo。

```
# lilo
```

12) 重新启动。

【提示】目前，对 **i8xx** 芯片的 **FrameBuffer** 设备驱动程序支持，已经包含在了最新的 **Linux** 内核 **2.6.x** 当中了。如果您有兴趣，也可以尝试使用最新的 **Linux 2.6.x** 内核来支持 **i8xx** 现实芯片。

附录 E MiniGUI 和嵌入式操作系统的关系

E.1 MiniGUI 和嵌入式操作系统的关系

为什么 MiniGUI 能够在如此众多的嵌入式操作系统上运行？这是因为 MiniGUI 具有良好的软件架构，通过抽象层将 MiniGUI 上层和底层操作系统隔离开来。如图 E-1 所示，基于 MiniGUI 的应用程序一般通过 ANSI C 库以及 MiniGUI 自身提供的 API 来实现自己的功能；MiniGUI 中的“可移植层”可将特定操作系统及底层硬件的细节隐藏起来，而上层应用程序则无需关系底层的硬件平台输出和输入设备。

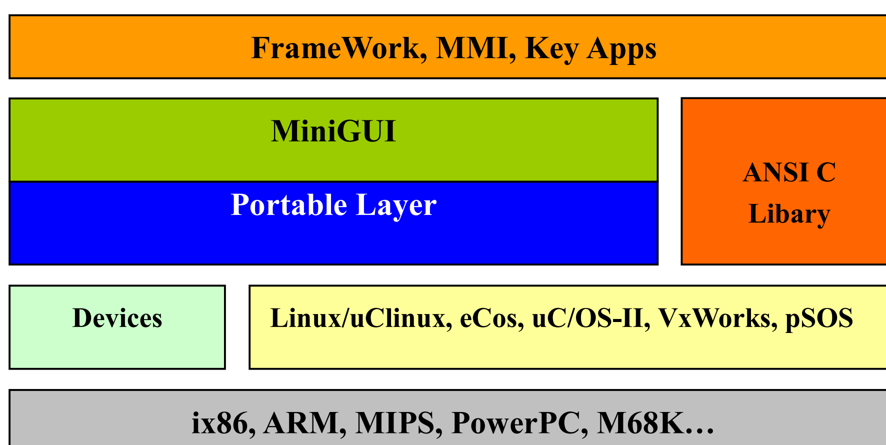


图 E-1 MiniGUI 和嵌入式操作系统的关系

另外，MiniGUI 特有的运行模式概念，也为跨操作系统的支持提供了便利。有关 MiniGUI 运行模式的解释，可参阅本手册附录 A。一般而言，MiniGUI-Standalone 模式的适应面最广，可以支持几乎所有的操作系统，甚至包括类似 DOS 这样的操作系统；MiniGUI-Threads 模式的适用面次之，可运行在支持多任务的实时嵌入式操作系统，或者具备完整 UNIX 特性的普通操作系统；MiniGUI-Lite 模式的适用面较小，它仅适合于具备完整 UNIX 特性的普通操作系统。不论采用哪种运行模式，MiniGUI 为上层应用软件提供了最大程度上的一致性；只有少数几个涉及初始化的接口在不同运行模式上有所不同。

下面我们以 uC/OS-II 操作系统为例，简单介绍 MiniGUI 到该操作系统上的移植。

E.2 MiniGUI 在 uC/OS-II 操作系统上的移植

uC/OS-II（http://www.micrium.com）是一个应用于实时嵌入式应用的开源、实时操作系统内核。因为 uC/OS-II 非常简单，只要有一个普通的 C 编译器，就能完成编译并运行该操作系统，因此，uC/OS-II 首先在教学中得到了广泛应用。因为其简单及实时性好的

特点，现在也有一些用户开始使用 uC/OS-II 操作系统开发正式的嵌入式产品。

将 MiniGUI 移植到 uC/OS-II 是相对复杂的一项工作。因为 uC/OS-II 中缺乏象 malloc/free 甚至是 printf/fprintf/sprintf 这样的接口，不过，最重要的还是缺乏与 POSIX-Threads 兼容的接口。为此，我们编写了 POSIX-Threads 的绕转接口，这些绕转接口兼容于 POSIX-Threads。同时，我们还实现了 malloc/free，以及 printf/sprintf/fprintf 等接口。

因涉及到如此多底层特性的增强和修改，整个系统的调试将是非常困难的。为解决调试问题，我们首先在 SkyEye（清华大学陈渝主持的自由软件项目，http://www.skyeye.org）模拟器上运行 uC/OS-II 操作系统及 MiniGUI for uC/OS-II。我们首先使用了 MiniGUI 内部的 Dummy GAL 引擎和 Dummy/Auto IAL 引擎（这两个引擎分别通过软件方法来模拟实际的输出和输入设备，比如 LCD 显示屏及键盘）来运行 MiniGUI 的应用程序。尽管我们看不到真实的屏幕输出，但从应用程序在运行过程中打印的输出信息，我们可以看到 MiniGUI for uC/OS-II 已经正常工作了。接下来的工作就是将 MiniGUI for uC/OS-II 移植到正式的硬件产品中。

我们在 S3C2410 开发板上用 ADS 的 armcc 编译并测试了 MiniGUI 的 uC/OS-II 版本。这次，我们针对这个开发板编写了正式的图形和输入引擎。MiniGUI 的所有示例程序都可以在这块开发板上正常运行。

MiniGUI for uC/OS-II 版本为 uC/OS-II 用户提供了另外一个很好的 GUI 解决方案。相对 Micrium.com 提供的 uC/GUI 而言，MiniGUI for uC/OS-II 版本能够提供更好的功能和特性。

附录 F 几种常用的图形引擎和输入引擎的配置

需要注意的是下面讲述的仅仅是 MiniGUI.cfg 在不同情况下的配置，如果您的系统想使用 qvfb 或者 vga16 等图形引擎，要确保在编译 MiniGUI 函数库的时候已经将相应的图形引擎包含进去了。

■ 默认的配置

```
[system]
gal_engine=fbcon
ial_engine=console

[fbcon]
defaultmode=640x480-16bpp
```

■ 运行在 qvfb 之上的配置

```
[system]
gal_engine=qvfb
ial_engine=qvfb

[qvfb]
defaultmode=640x480-16bpp
display=0
```

■ 运行在标准 VGA 16 色之上的配置

```
[system]
gal_engine=vga16
ial_engine=console
```

附录 G 关于 MiniGUI 运行环境的裁减

关于 MiniGUI 的裁剪，我们的做法一般遵循如下的步骤：

1) 先根据自己目标环境的需要，确定自己需要 MiniGUI 的哪些资源。比如，是否需要 12 点阵字体，是否需要 16 点阵字体，需要等宽字体还是变宽字体。这样就可以将 MiniGUI.cfg 文件中 rawbitmapfonts、varbitmapfonts、qpf、truetypefonts、type1fonts 段中不需要的项删除掉，同时将 font_number 修改成所需要的字体的数量。然后到 /usr/local/lib/minigui/fonts 目录中，将那些不需要的字体资源文件删除，对于您的不同的项目需求，可以裁减到 200K 多一点，也就是该目录下中只剩下 2 个文件。字体资源目前是最影响存储空间占用的部分。

2) 同样，修改 MiniGUI.cfg 文件中的 imeinfo 段中的 imenumber，可以将五笔等输入法去掉。不要忘记同时将对应目录（/usr/local/lib/minigui/imetab）中的五笔输入法模块、双拼、自然码等文件删除掉。

3) 还是一样，修改 MiniGUI.cfg 文件中的关于 cursorinfo、iconinfo、bitmapinfo 等字段，并删除对应目录（默认为 /usr/local/lib/minigui/res）下的对应文件做到这一步，MiniGUI 的资源大小就可以裁减到 600K 以内。需要注意的是，这些文件在您安装 MiniGUI 资源时，执行 make install 之后就已经安装到系统上了，所以要自己手动删除上面提到的文件。

4) 在用户安装 MiniGUI 函数库时，执行 make install 之后，MiniGUI 的函数库默认安装在路径 /usr/local/lib 中。此时函数库的大小为动态库 2200K，静态库 6.5M。不过一般情况下是不需要静态库的。可以用 strip 命令删除 MiniGUI 的函数库和用户程序中的符号信息，这样函数库就可以减小到 400-600K 左右。此时，函数库和资源加在一起就在 1M 之内了。一般 Linux 系统也就是 500-800K。总共的环境就在 2M 之内了，再加上用户应用程序，系统总的空间可以控制在 2M 到 4M 之内。

现在我们来看看关于 MiniGUI 函数库的裁减问题。

从大的方面说，函数库分为 3 个部分：libminigui、libmgext、libvcongui。其中第一个 libminigui 是必需的。第二个函数库 libmgext 大概 100K 左右，如果您不使用 MiniGUI 的月历控件、树型控件，动画 GIF 支持等扩展函数库支持的话，也可以删除这个函数库。第三个函数库 libvcongui 是 MiniGUI 的虚拟控制台函数库，一般来说只要您不调用虚拟控制台程序也是不需要的。

从小的方面说，在编译函数库的时候，还可以使用./configure --disable-big5support --disable-type1support 禁止函数库对 big5 编码和 type1 矢量字体的支持。但是这些工作基本是使函数库尽量少包含一些代码，使函数库在尺寸上减小 10K 左右，对于裁减来说帮助不大。

附录 H 常见问题及解答

H.1 授权问题

1．MiniGUI 采取何种授权方式？

答：MiniGUI 是遵循 GNU 通用公共许可证（GPL）发布的自由软件，但亦可使用在专有的商业软件产品中，授权详解请参考附录 I。

2．我使用 MiniGUI 需要向飞漫软件技术有限公司支付授权费用吗？

答：MiniGUI 的用户或者爱好者可以通过飞漫软件的网站获得 MiniGUI 的源代码，只要用户遵循 GPL 条款的精神使用 MiniGUI，就无需为使用 MiniGUI 而向飞漫软件支付授权费用，但这要求您基于 MiniGUI 的应用程序也必须遵循 GPL 条款发布；如果您的专有或商用软件产品使用了 MiniGUI，而又不想遵循 GPL 条款发布您的应用软件，则应该向 MiniGUI 的版权拥有者北京飞漫软件技术有限公司支付商业授权费用。飞漫软件就 MiniGUI 的授权策略见本手册附录 I。

3．我们想采用 MiniGUI，但希望贵公司能够提供软件质量的担保，这如何做到？

答：根据 GPL 条款的定义，以 GPL 条款发布的软件是绝对不含任何担保的，包括连带或者隐含的担保。但这并不意味着自由软件的质量得不到任何保证，相反，诸如 MiniGUI 这样的自由软件正在许多场合发挥着稳定的、可以依赖的作用。如果您要获得 MiniGUI 的软件质量担保，包括软件缺陷修正，免费升级等等，则可通过购买 MiniGUI 商业授权的方式获得。

4．哪些行为会侵犯飞漫软件的合法权益？

答：飞漫软件作为多个自由软件项目的版权拥有者，以源代码方式发布这些自由软件，其目的是为了给用户了解软件内部机制，并根据需求进行适当定制的自由和方便。但是，由于大多数用户对 GPL 条款的不了解，有时会无意中出现违反 GPL 条款的情况，这对商业用户尤其不利。这些行为主要有：

- 盗用自由软件的部分或全部代码并使用在其他场合；更甚者，将 MiniGUI 等自由软件据为己有，改头换面，以私有软件形式出售。这种行为属于严重侵权行为，自由软件的版权拥有人会追究这种侵权行为，侵权人将面对严重的刑事和民事处罚。
- 对自由软件源代码进行了修改或者增强，并用于商业目的，但并未遵守 GPL 条款

公开修改后的源代码。常见情况：修改 MiniGUI 某些控件的实现以满足其需求；修改 MiniGUI 的输入法实现以满足其需求等等。

- 按照 GPL 条款，采用 MiniGUI 开发图形用户界面应用程序也应该遵循 GPL 条款发布。如果您使用 MiniGUI 的应用程序既没有遵循 GPL 条款发布，也没有购买 MiniGUI 的商业授权，则这种行为亦属于典型的软件盗版行为。

H.2 一般性问题

5．有哪些使用 MiniGUI 的产品成功上市？

答：在手持终端领域、数字控制领域、POS 机/彩票机、电力控制以及信息终端等领域，均有使用 MiniGUI 成功上市的产品。遗撼的是，许多产品并不是类似掌上电脑这样的大众化消费类产品，因此，非专业领域的人较难接触到这些产品。有关这些产品的介绍，您可以到 http://www.minigui.com/project/cindex.shtml 上查阅。

6．MiniGUI 的稳定性如何？

答：这个问题很难回答，因为有时引起系统不稳定的因素可能在应用程序，而不是底层函数库。不过，我们可以给您提供一些数字供参考：

- 在一个比较复杂的 MiniGUI 应用程序中，模拟用户的按键操作进行多个窗口之间的切换测试，连续测试 2 天共约 100,000 次按键不会出现任何问题。

- 架构在 MiniGUI 之上的许多工业控制系统（有的甚至要求达到毫秒级的实时性），已经应用于真正的工业场合，并稳定运行。

7．MiniGUI 本身占用多少内存？

答：MiniGUI 占用的内存和您在配置 MiniGUI 时使用的选项有关，也和 MiniGUI 配置文件中指定的运行时资源的装载量有关。如果使用最小的资源装载量，在使用 Linux 内核 2.0 时，MiniGUI 可以在只有 4M 内存的系统上运行。如果使用 Linux 内核 2.2 或者 2.4，这个内存需求要增加 2 MB。

8．MiniGUI 占用多少存储空间？

答：MiniGUI 的函数库和资源所占用的存储空间与您在配置 MiniGUI 时使用的选项有关，也和 MiniGUI 配置文件中指定的运行时资源的装载量有关，而影响资源大小的关键是字体资源。在使用较小字体资源的情况下，MiniGUI 的函数库和资源占用的存储空间小于 2 MB。每使用一款多字节字符集的字体（比如中文简体、日文或者韩文字体），平均要增加 200

KB 到 600KB 的空间。

H.3 可移植性问题

9．MiniGUI 是否支持 uCLinux？

答：现已证明，MiniGUI 1.3 以前的版本未经修改即可在 uClinux 操作系统上运行。MiniGUI 1.3 及其后版本针对 uClinux 操作系统进行了定制和优化，性能更加出色。

10．MiniGUI 在哪些嵌入式 CPU 上成功运行过，最低的 CPU 主频大概是多少？

答：MiniGUI 在含有 MMU 单元的 ARM 系列 CPU（比如 StrongARM、xScale 等），PowerPC、MIPS 系列 CPU（比如 EP7312、VR4181 等）、无 MMU 单元的 m68k 系列 GPU 上都有过成功运行的实例。在这些 CPU 中，最低的 CPU 主频大约为 16 MHz，CPU 运算能力为 10 MIPS。

11．MiniGUI 能支持单色 LCD 显示屏吗？

答：可以。MiniGUI 能支持从单色、四色、十六色到 256 色、4096 色、65536 色等几乎所有颜色深度的显示屏。

12．MiniGUI 可在多大的显示屏上正常运行？

答：理论上讲，MiniGUI 的正常运行不受显示屏大小的限制。

13. MiniGUI 可以在 VxWorks、uC/OS-II、eCOS 等嵌入式操作系统上运行吗？

答：MiniGUI 可在上述操作系统上运行，针对这些操作系统的支持包含在 MiniGUI 1.6.x 以上版本中。

H.4 配置问题

14．哪种情况下应该使用 LibGGI 或者 SVGALib 作为 MiniGUI 的图形引擎？

答：如果您的 Linux 系统能够支持 FrameBuffer 设备，则应该选择 MiniGUI 内建的 NATIVE/FBCON 图形引擎，这样就没有必要安装 LibGGI 或者 SVGALib。SVGALib 是一个老的图形库，不需要内核 FrameBuffer 的支持，可以支持一些老的 SVGA 显示卡。如果您的 Linux 内核中没有 FrameBuffer 驱动程序，则只能选择 SVGALib，比如在选用 Linux 2.0.x 为系统内核的情况下。LibGGI 是一个比较好的图形库，运行在 LibGGI 上的

程序可以在 X Window 下运行。这样，如果使用 LibGGI 作为 MiniGUI 的图形引擎，就可以在 X Window 上运行 MiniGUI，从而方便程序的调试。但要想在 X Window 上运行 MiniGUI，还有一个更好的选择就是 qvfb 引擎。因此，除非有一些特殊需求，否则没有必要安装 LibGGI 或者 SVGALib。

15．哪种情况下应该使用老的 GAL 接口？

答：MiniGUI 的 NEW GAL 以及建立在 NEW GAL 之上的 NEW GDI 接口提供了非常丰富的功能，但是，NEW GAL 接口要求底层图形设备采用线性 FrameBuffer 结构，而且其颜色深度应该在 8 位色及以上。因此，如果您要支持低于 8 位色（256 色）的显示设备，就应该采用老的 GAL 接口。

另外，对类似 COMPAQ iPAQ 的设备，其显示屏的逻辑坐标和物理坐标相差 90 度，MiniGUI 提供了坐标转换功能。但目前这一功能是在老的 GAL 接口基础上实现的。因此，如果需要这个坐标转换功能，就需要使用老的 GAL 接口。

16．我的显示卡只支持标准 VGA 16 色模式，我该如何配置 MiniGUI？

答：首先使用下面的选项进行配置：

```
./configure --disable-lite --disable-newgal --enable-vga16gal
```

编译安装之后，修改 /usr/local/etc/MiniGUI.cfg 文件：

```
[system]
gal_engine=VGA16
ial_engine=console

[VGA16]
defaultmode=G640x480x16
```

然后就可以正常运行 MDE 中的演示程序了。

H.5 编译问题

17．我打开了 MiniGUI 的 TrueType 字体支持选项，为什么编译时出现了许多错误？

答：这是因为您系统中的 TrueType 字体支持库 libttf 版本太高的原因造成的。MiniGUI 使用 LibTTF 1.3.1 版本。在 RedHat Linux 6.2 等某几个特定的发行版系统中，默认安装的是 LibTTF 2.0 版本的函数库，并且未进行版本差异处理，因此会由于版本差异而导致编译时出现大量问题。这时，您可以安装 LibTTF 的 1.3.1 版本，或者干脆使用 --disable-ttfsupport 选项禁止 TrueType 字体支持。

18．为什么在编译函数库的时候，有时会出现

```
    can not make hard link filename.o to filename.lo.
```

这样的错误？

答：符号链接和硬链接是 UNIX 文件系统中才具有的文件类型。如果您在非 UNIX 文件系统上编译由 Automake/Autoconf 脚本维护的函数库，则无法建立硬链接或符号链接。请检查您的文件系统，是否是 FAT32 之类的文件系统。

H.6 运行时问题

19．为什么我运行 MDE 包中的程序时，会告诉我

```
    AttachSharedResource: No such file or directory
    Error in step 6: Can not attach shared resource!
    Initialize minigui failure when using /etc/MiniGUI.cfg as cfg file.
```

答：如果您的 MiniGUI 被配置为 MiniGUI-Lite，则应该首先运行 mginit 程序。MiniGUI-Lite 版本基于客户/服务器体系运行，在运行客户程序之前，必须启动一个名称为 mginit 的服务器程序。在 MDE 包中，首先运行 mginit/ 目录下的 mginit 服务器程序，然后就可以运行其它目录下的演示程序了。

20．我在运行 MiniGUI 时，遇到如下提示信息：

```
    GAL ENGINE: Error when opening /dev/fb0: Permission denied. Please check your ker
nel config.
    GAL: Init GAL engine failure.
    Error in step 3: Can not initialize graphics engine!
    Initialize minigui failure when using /usr/local/etc/MiniGUI.cfg as cfg file
```

为什么会出现这样的情况？

答：这是因为您的操作系统尚未激活 FrameBuffer 驱动程序，或者，您的 /dev/fb0 设备的访问许可不正确。 请参阅本手册附录 C 的 C.1 一节正确配置并激活 FrameBuffer 驱动程序。

21．我在运行 MDE 的 mginit 时，出现如下错误消息：

```
    Error in step 2 : There is already an instance of minigui.
    Initialize minigui failure when using /usr/local/etc/MiniGUI.cfg as config file.
```

是怎么回事？

答：这种情况一般有两个原因。第一，您已经运行了一个 mginit 程序。第二，您上次运行 mginit 时没有正常退出系统。如果是第二个原因，您只需删除 /var/tmp/ 目录下的

minigui 和 mginit 两个文件即可。如果还有错误，请重新启动计算机。

22．我已经按照附录 C 的 C.1 一节的说法正确配置了 VESA FrameBuffer，但为什么我的显示模式还是字符的？

答：这是因为某些显示卡，比如 Intel 的 i810 系列显示卡，本身不是 VESA 兼容的，所以，Linux 内核的 VESA FrameBuffer 并不能够提供对这种显示卡的支持。尽管您正确配置了 VESA FrameBuffer，但仍然只是字符模式。

23．Linux 内核中有没有对 Intel i810 显示卡的 FrameBuffer 驱动程序支持？

答：目前，Linux 正式发行的内核当中还没有对 Intel i810 显示卡的 FrameBuffer 驱动程序支持。但您可以根据本手册附录 D 的指导进行配置。除了 VESA FrameBuffer 驱动程序以外，Linux 2.4.x 内核中包含了对 ATI Mach64、ATI Rage128、3Dfx Banshee/Voodoo3、SIS 630/540 等等显示卡的支持。由于大多数 PC 显示卡都是 VESA BIOS 2.0 兼容的，所以利用 VESA FrameBuffer 驱动程序可以支持最广泛的 PC 显示卡。

24．在 PC 上，MiniGUI 到底支持哪些鼠标类型？

答：MiniGUI 在 PC 上运行时，目前所支持的鼠标协议有 MS、MS3、PS2、智能 PS2（IMPS2）等常见的鼠标协议。目前的 MiniGUI 还不支持智能鼠标上的滚轮。

25．我想支持老式的串口鼠标，该怎么办？

MiniGUI 可通过 GPM 支持几乎所有的鼠标类型。配置过程如下：

1) 运行 gpm -k 命令杀掉正在运行的 gpm。
2) 运行 mouse-test 命令确认自己的鼠标设备和协议。
3) 运行 gpm，指定鼠标设备和协议：

```
gpm -R -t <yourmousetype> -m <yourmousedevice>
```

4) 编辑 MiniGUI.cfg 文件，将 mtype 设置为 gpm；将 mdev 设置为 /dev/gpmdata：

```
[system]
...
mtype=gpm
mdev=/dev/gpmdata
```

然后启动 MiniGUI 就可以了。需要注意的是，用 gpm 设置鼠标格式的时候，可以使用 –R 参数，gpm 的 -R 参数可用来将原有鼠标协议转换成 GPM 所定义的鼠标协议，并出现在 /dev/gpmdata 文件上。

26．在执行 MDE 的 mgint 时显示错误信息：

```
NEWGAL: Does not find matched engine: fbcon.
Error in step 3: Can not get graphics engine information!
```

是什么原因？

答：这是 NEWGAL 接口的 FBCON 引擎在初始化 FrameBuffer 设备时失败而导致的问题，通常是因为您的内核不支持 FrameBuffer 驱动程序，或者没有激活 VESA FrameBuffer 驱动程序，或者您没有适当的访问许可打开 /dev/fb0 设备造成的。请参阅本手册附录 C 的 C.1 小节或检查 /dev/fb0 设备文件的访问许可设置。

27．在执行 MDE 的 mginit 时显示错误信息：

```
vesafb does not support changing the video mode
```

是什么意思？

答：这句话是一条警告消息，可以忽略。它是针对 VESA FrameBuffer 驱动程序而言的。VESA FrameBuffer 驱动程序不支持运行过程中的显示模式切换，而只能通过内核的引导选项来确定显示模式。一旦确定，就无法改变，除非修改引导选项并重新启动系统。

28．在执行 MDE 的 mginit 时显示错误信息：

```
NEWGAL: No video mode large enough for the resolution specified.
NewGAL: Set video mode failure.
```

是什么意思？

答：这是因为您在 MiniGUI.cfg 中指定的显示分辨率比 FrameBuffer 能达到的显示分辨率大。您可以试着修改 MiniGUI.cfg 文件指定一个较小的显示分辨率。

29．如何才能关闭 MDE 程序启动之后的输入法条？

答：MDE 程序启动时的输入条可以用左边的 <Ctrl> 键切换；另外，也可以通过配置 MiniGUI.cfg 文件里的 imenumber 选项：

```
[imeinfo]
imenumber=2
```

将 imenumber 设置为 imenumber＝0，启动时就不会出现输入条了。

30．怎么样把 MiniGUI 的运行画面保存为图片？

答：在运行 MiniGUI 程序时，按 <PrntScrn> 键就会在当前目录保存整个屏幕的 BMP 文件，文件名为 0-<NO>.bmp，其中 <NO> 是按键次数的编号；按 <Ctrl+PrntScrn> 键，可保存当前活动主窗口的 BMP 文件，文件名为 <HWND>-<NO>.bmp，其中 <HWND> 是

活动主窗口的句柄，<NO> 是按键次数编号。

31．我运行 MDE 的 mginit 程序时，为什么显示两个对话框之后程序就退出了？

答：这通常是因为您编译安装的 MiniGUI 不含 PNG 图像文件的支持功能导致的。在某些 Linux 发行版上（比如早期的 TurboLinux 版本），因为所含的 PNG 图像支持库（即 libpng）版本太早，在进行 MiniGUI 配置时自动禁止了对 PNG 图像的支持而导致的。这种情况下，MiniGUI 的 LoadBitmapFromFile 函数无法正确装载 PNG 图像文件，而 MDE 的 mginit 在启动时恰好要装载两个 PNG 图像文件，这样，由于无法装载图像文件，mginit 程序退出。

解决这个问题的办法有两个。第一，您可以从网上下载安装最新版的 libpng 库；第二，修改 mginit.rc 文件中 [mginit] 段的 nr 键值，使之小于 8。

另外一个可能导致这个错误的原因是，您没有在 mginit 程序所在的目录启动这个程序。请使用 cd 命令改变到 mginit 文件所在的目录，然后运行该程序。

32．在 MiniGUI-Lite 中，我如何从 MiniGUI 切换到其它控制台？

答：在 MiniGUI-Lite 中，可以按 <Right_Ctrl + Fx> 组合键，从 MiniGUI 中切换到其它虚拟控制台。另外，您还可以通过 <Ctrl+Alt+Backspace> 组合键强制退出 MiniGUI。MiniGUI-Threads 目前还不提供类似的功能。

附录 I MiniGUI 授权策略

MiniGUI 1.3 和 MiniGUI 综合演示程序 MDE 遵循 GUN 通用公共许可证（简称 GPL）发布。飞漫软件针对 MiniGUI 的授权策略概括而言就是：MiniGUI 是 100% 按照 GPL 条款发布的自由软件，如果用户能 100% 遵守 GPL 许可证条款，则无需支付任何授权费用。在其他任何情况下，都需要获得飞漫软件技术有限公司的商业授权。

老版本 MiniGUI（1.3.0 版本之前）遵循 LGPL 条款发布。因为嵌入式系统无法保障用户获得修改 MiniGUI 函数库并进行调试的自由，从而无法确保 LGPL 在嵌入式产品中的实施，因此，如果要在嵌入式产品中使用老版本的 MiniGUI， 也必须首先购买商业授权。

为了确保对 GPL 条款 的正确理解，本附录第三小节给了从 GNU 网站获得的英文条款原文（因为任何经过翻译的文本均有可能无法精确表达该条款的原意）。用户，尤其是打算将自由软件用于商业产品中的用户，必须仔细阅读并理解这些自由软件条款。

I.1 授权详解

自 MiniGUI V1.3.0 起，飞漫软件技术有限公司遵循 GNU General Public License 许可证（简称 GPL）发布 MiniGUI。该条款的原文可见和 MiniGUI 源代码一同发布的 COPYING 文件。相比早期版本使用的 LGPL（GNU Lesser General Public License）许可证，GPL 所定义的条款要严格的多，更能有效维护自由软件的权益，避免自由软件成为专有系统的一部分。

飞漫软件技术有限公司为无法或者不愿 100% 遵循 GPL 条款使用 MiniGUI 的用户提供商业授权。具体价格及购买方法，请访问飞漫软件网站

```
http://www.minigui.com/product/cindex.shtml。
```

1. 如果您 100% 遵循 GPL，无需获得商业授权

如果您使用 MiniGUI 的应用程序以 GPL 发布，则无需获得我们的商业授权。我们非常欢迎任何人在遵循 GPL 条款的基础上复制、修改和发布 MiniGUI。在这种情况下，您无需获得飞漫软件的任何形式的（包括口头或书面）使用授权，因为 GPL 条款本身就足够确保您的权益。但需要注意的是，飞漫软件不对这种形式下的使用提供任何形式的担保或技术支持。

2. 如果您从不复制、修改和发布 MiniGUI，无需获得商业授权

只要您从不复制、修改和发布 MiniGUI，则您可以在您的应用程序中使用 MiniGUI，

而无需获得商业授权。举个例子，您在完成一篇学位论文，并因此在您的程序中使用了 MiniGUI，您的程序仅仅用来说明您论文中试验的可行性或者结果，而且您没有修改 MiniGUI，并且该程序不以任何方式被复制和发布，则您无需获得商业授权。飞漫软件也不对这种形式下的使用提供任何担保。

但需要特别指出：

- 修改－－我们欢迎您对 MiniGUI 进行任意的修改。如果你发布该修改版本，则您对 MiniGUI 所做的任何修改、所有的接口代码以及直接和间接地与接口相关联的代码将遵循 GPL 许可证。
- 复制。我们允许您复制 MiniGUI 二进制代码和/或源代码，但一旦这么做了，所有的副本应遵循 GPL 许可证。

3. 其他情况均需获得商业授权

如果您使用 MiniGUI 的应用程序并不以 GPL 条款发布，却打算在内部或外部发布使用 MiniGUI 的应用程序或者函数库，则您必须首先获得飞漫软件的商业授权。

特别是：

- 您的非 GPL 应用程序连接了 MiniGUI，不管静态还是动态连接，您需要为每一个 MiniGUI 函数库副本购买商业授权。
- 如果您在自己的单位使用 MiniGUI 函数库，但又不希望将其置于 GPL 许可证之下，则需要购买商业授权。
- 当然，更多的人购买 MiniGUI 的商业授权，其目的非常简单，他们希望获得来自飞漫软件的技术支持和软件质量担保。

4. 建议

对商业用户，我们建议购买 MiniGUI 的商业授权。这不仅仅能帮助您避免为满足 GPL 条款而付出太多的硬件和软件开发费用，从而保护自己专有系统的商业利益，也可以从飞漫软件获得质量担保——GPL 软件不含任何形式的、间接或直接的担保。

对自由软件社区的用户，或者经费不足的科研院校，我们建议您在开发基于 MiniGUI 的应用程序时，采用 GPL 或者其他的开放源码许可证条款。这样，能在最大程度上满足 GPL 许可证条款，您也不必购买 MiniGUI 的商业授权。

如果您不知道自己的产品能否 100% 满足 GPL 条款，则建议您选择商业授权，因为对飞漫软件来讲，我们会保护商业客户的利益，并通过优秀的技术支持和产品担保来确保您的产品能够顺利开发并良好运行。

5. MiniGUI 商业授权方式

飞漫软件提供以出货量为单位的 MiniGUI 商业授权办法，用户可以根据自己产品的出货量灵活选择：

- 用户需要为每个 MiniGUI 副本购买商业授权，每个授权的单价和使用 MiniGUI 的产品数量有关。

另外，商业授权费用与您使用的操作系统（Linux 或者 uClinux）有关。具体的价格及购买方法，请访问飞漫软件网站

```
http://www.minigui.com/product/cindex.shtml
```

I.2 老的版本

在版本 1.3.0 之前，MiniGUI 使用 LGPL 条款发布。这种条款为非自由软件使用自由函数库而提供了非常宽松的条件。但是，LGPL 条款仍然为复制、修改和发布 LGPL 软件定义了一些约束性的条款，以避免 LGPL 软件变成专有系统事实上的一部分，或者通过一些技术障碍来阻止用户获得 LGPL 条款定义的自由。这些条款包括以下几个方面：

- 对 MiniGUI 本身的复制、修改和发布行为，均应在确保 MiniGUI 仍然为函数库、仍然遵循 LGPL 或者 GPL 许可证的前提下进行。
- 使用 MiniGUI 必然要生成可执行文件。根据 LGPL 条款的定义，该可执行文件是 MiniGUI 的“衍生作品”，并且应该按照 LGPL 许可证之第 6 条发布该可执行文件。该条款的核心思想是，确保用户知悉在该“衍生作品”中使用了遵循 LGPL 条款发布的 MiniGUI 函数库，用户因此将得到修改 MiniGUI 的权利；在用户修改了 MiniGUI 函数库的情况下，只要修改后的版本和原先的版本在接口上是兼容的，则应确保用户仍能够生成“衍生作品”（静态连接的情况），或者“衍生作品”仍然能够正常 工作（动态连接的情况）。因此，“衍生作品”的发布必须以确保用户获得上述自由为前提和条件。

飞漫软件认为，MiniGUI 在大多数嵌入式系统中的应用，会因为某些技术上的障碍而不可避免地阻碍用户获得上述自由：

- 某些嵌入式系统根本不存在任何硬件上的条件或机制（比如程序上载接口），以帮助用户运行和调试修改后的 MiniGUI 函数库。
- 某些嵌入式系统采用了其他非开放源码的专有操作系统，用户根本无法免费获得用于编译、连接和调试修改后 MiniGUI 函数库的工具。
- 某些嵌入式系统，因为专利或技术保密等原因，禁止用户对程序采用反向工程和反汇编，从而禁止用户调试修改后的 MiniGUI 函数库。

如果您剥夺了用户获得修改 MiniGUI 的权利，即使您遵循了 LGPL 许可证的其他条款发布了修改后的 MiniGUI 源代码，则仍不属于 100% 遵循 LGPL 条款。这时，您需要获得飞漫的商业授权。那么，嵌入式系统需要完成哪些工作才算 100% 遵循 LGPL 条款呢？

- 确保按照 LGPL 条款复制、修改和发布 MiniGUI 函数库本身。
- 如果您采用静态连接方式生成使用 MiniGUI 函数库的可执行程序，则应确保提供用于生成最终可执行程序的全部目标代码和/或源代码。
- 如果您采用动态连接方式使用 MiniGUI 函数库，则应确保在达到接口兼容性的情况下，使用 MiniGUI 函数库的可执行程序仍然能够正常工作。
- 您必须允许用户对您的程序进行反向工程，以便用户调试修改后的 MiniGUI 函数库。
- 如果您采用了非开放源码的专有嵌入式操作系统，则请确保和您的产品一同提供用于编译、连接和调试程序的工具，或者您的用户可免费获得这些工具。
- 您的产品还应该在硬件和软件上提供替换原先 MiniGUI 函数库和/或使用 MiniGUI 的可执行程序的机制和条件，以便用户在您的嵌入式产品中调试和运行修改之后的 MiniGUI 函数库。
- 其他为确保用户具有修改 MiniGUI 函数库的自由而应该提供的法律许可及软硬件条件。

因此，我们认为在嵌入式产品中使用老版本 MiniGUI 也必须购买商业授权。

I.3 GNU 通用公共许可证

```
           GNU GENERAL PUBLIC LICENSE
              Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                      59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

               Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.
```

```
  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

           GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:
```

```
    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
```

```
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.
```

```
  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.

  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

              NO WARRANTY

  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

            END OF TERMS AND CONDITIONS

        How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
```

```
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA


Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```